Author:
Sherfield, Chris

Title:
Game theory applied to cybersecurity threat mitigation - Analysis of Threshold FlipThem

Game theory applied to cybersecurity threat mitigation

Analysis of Threshold FlipThem

By

CHRISTOPHER SHERFIELD

Department of Computer Science
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of Engineering.

Abstract

A standard method to protect data is to apply threshold cryptography in the form of secret sharing. This is motivated by the acceptance that adversaries will compromise systems at some point; hence using threshold cryptography provides a defence in depth.

The existence of such powerful adversaries has also motivated the introduction of game theoretic techniques into the analysis of systems, e.g. via the FlipIt game of van Dijk et al. This work further analyses the case of FlipIt when used with multiple resources, dubbed FlipThem in prior papers. We define key extensions of the FlipThem game as a customisable framework where the attacker's goal is to compromise a threshold of the resources, a game we aptly name Threshold FlipThem.

We introduce the single-rate version of the game which restricts the number of rates to just one per player. Two forms of reset are considered based on how many resources are flipped in one move. Another consideration is separate costs and strategies for each resource. We calculate analytic benefit functions based on the rates and costs of the players. From these, equilibria of the game are found for the benefit functions and conditions calculated to see when these equilibria are valid.

Next, we consider two learning approaches. Fictitious play is introduced in which players do not know opponent costs, or assume rationality. Each player responds to the observed actions of the other player over a continuing sequence of epochs, instead of calculating an equilibrium. In our final form of learning, we remove the assumption that players know analytically their payoff functions and move costs and use genetic algorithms. Populations of strategies are evolved over many iterations to find optimal strategies within multiple strategy populations.

We introduce the FlipThem simulation lab, a python framework designed to create and test strategies within the game of Threshold FlipThem. This is offered as open-source software, allowing researchers to explore their own defined strategy classes.

Dedication and acknowledgements

I would like to begin by thanking my two supervisors Nigel Smart and David Leslie for their unfaltering guidance and supervision throughout this doctorate. Their separate areas of expertise left no hole of despair for me to fall into.

This work would not have been possible without the financial support of the Engineering and Physical Sciences Research Council and GCHQ; thank you for allowing me to undertake this PhD.

I would like to thank my parents for begrudgingly accepting my seemingly eternal student status. They have never failed to be the most supportive and encouraging parents I could have ever hoped for.

Thank you to Kirsty for enduring my irrational annoyances, grumps and whines. Studying together made the most boring topics fun. For that, I am forever grateful.

I would like to thank the Bristol crypto group for sitting through my game theory talks and always asking interesting questions. A special thanks to Ana, for always alerting me when there was food in the office!

Thank you to James, who took the time to review earlier versions of this manuscript and for always having an answer to my grammatical nightmares, even at 3am. Finally, I could not have made it through 3.5 years of research without the amazing outlet that is the University of Bristol volleyball club. Thank you for allowing me to be a part of such an awesome society.

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

TABLE OF CONTENTS

List of Tables

Table **Page**

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

At the heart of cryptography is a secret; a secret that despite thousands of hours of research, manpower and bloodshed, it cannot be found. Nothing is safe. Just as in the physical world where belongings can be taken, in the cyber world a person's money, data, and even identity can be ruthlessly stripped away in the blink of an eye leaving no trace of evidence to follow.

Whilst the above paragraph has been written in an elaborate and dramatic manner in an attempt to catch the reader's eye, it is not wholly untrue. Cryptographic schemes are continually having to be reworked, manipulated, or even scrapped due to the ongoing work of hackers (both ethical and unethical), who find increasingly ingenious ways to break into supposedly secure systems. It is the role of this thesis to assume that these systems are almost surely fallible and to make decisions on how best to defend cyber systems based on the workings of game theoretical models.

According to Herley [11], over two billion people use the internet, and yet an extremely small percentage are victims of cyber attacks. Why is this? Herley splits up attacks into two different types; targeted and non-targeted (or scalable). The non-targeted can be scaled up for almost imperceptible cost in order to reach more people. However, because the non-targeted attacks are not specific to the user they are attempting to infiltrate, they can only be so powerful. Targeted attacks hit specific users due to their value, the users have been studied in order to be more successfully infiltrated. This severely lowers the ability to scale up the attack to target more users. With the amount of work put into one specific user being proportional to the amount of reward achievable, high value targets are chosen. Clearly, very few users have high enough value to warrant the effort, which explains why the regular, or average, user can succeed with general internet use (email, banking etc.) without being compromised; it just is not worth being attacked.

The traditional methodology of securing systems is to rely solely on cryptographic means to ensure protection of data (via encryption) and authentication (via signatures and secured passwords). However, both of these techniques rely on some data being kept secure. The advent of attacks such as Advanced Persistent Threats (APTs) means that exfiltration of the underlying cryptographic keys, or other sensitive data, is now a real threat. To model such long term stealthy attacks researchers have turned to game theory [1, 19, 22, 25], adding to a growing body of work looking at game theory in cybersecurity in various scenarios [5, 13, 30]. The main game theoretical model we discuss is the game of *FlipIt* and our own extensions and analysis of it. First, we discuss the motivations behind this game and through doing so the notion of fallibility we have so carelessly accused many world-renowned cryptographer's work of.

1.1 The power of adversaries

Reports of high profile cyber security attacks are increasing in number over time. We give examples of these relevant attacks now; in particular, the examples have been chosen purely based on the author's interest or because the incidents happened around the time of the writing of this work.

1.1.1 Real world attacks

In the three months leading up to 17th February 2017[1], there were 188 reported high-level cyber attacks. According to Ciaran Martin, chief executive of the National Cyber Security Centre (NCSC), some of these have been state sponsored attempts from countries such as Russia and China. Martin states that attempts on government departments are designed to extract classified data on various sectors across the UK government.

On the 23rd June, BBC News[2] reported on the British parliament's computer systems coming under attack from an adaptive brute force attack. Large systems such as these, which contain a lot of valuable and secure information, can be expected to be attacked often. However, parliament's security operations team started to notice the intensity of attacks increasing. The team of security experts began to fight back. With this increase in defence, the number of attacks responded in kind; around 200,000 attempts were made to enter parliamentarians' email accounts. The initial attack was a brute force style attack, targeting weak passwords, meaning the attackers were able to avoid their attempts being spotted. Once noticed, the parliamentary team began to block certain areas of the system by closing it down. The security team realised the hackers were reacting, changing their focus by attacking other services. It is estimated that around 50 email accounts were compromised. While it has not been confirmed who was behind the attack, the

[1] *http://www.bbc.co.uk/news/uk-38951172*
[2] *http://www.bbc.co.uk/news/uk-40619309*

security experts have strongly indicated that this sort of attack has come from an experienced, well funded adversary; most probably a state activity.

Whilst this work is in no way attempting to breach the boundary between academia and politics, and enter a "who did what" debate, I use these examples to show the importance of security and all aspects of it. We must understand that simply locking your front door with the most secure deadlock to date does not ensure complete security. We must expect failure and have a backup plan for that eventual failure. It is where this work begins its adventures into the wilderness of exploration and discovery. How do we combat the silent assassin? Should we accept that adversaries will always be in partial (or complete) control of our system?

1.1.2 Advanced Persistent Threats

As discussed previously, cyber attacks are a real threat and must be treated as such. However, we have been informal in our treatment of these attacks and this needs to be fixed. Over the past few years a new term has come to the forefront to describe and define these powerful attacks, Advanced Persistent Threats (APTs).

Advanced refers to both the adversary's skill level and their resource level, using a large array of techniques such as various reconnaissance and attack methods. However, note that it is often the adversary's method of entry into a network or system is not necessarily particularly advanced, using techniques such as phishing, social media or information gathering. Once the adversary has gained access to the targeted system, the methods used to remain undetected are advanced. For example, attackers have used malware code that constantly changes by recompiling itself. The term *persistent* in APT refers to the attempt of the attacker to remain in the system, gaining information for an extended period of time. In the case of non-APT attacks, once attackers are in a system, they initially attempt to cause havoc before leaving promptly, whilst all the while hoping not to be caught. In APTs, the adversary stays in the system, remaining undetected for as long as possible. This can mean they use a slow, stealthy process in order to not arouse suspicion. It is easy to see that the examples discussed above can be deemed APTs. Whilst we have not gone into depth about the subject, it can be seen that this is an extreme threat to the security of systems and needs to be taken seriously. The role of this work is to assume adversaries have a large amount of power and are able to break into a system at the push of a button. We abstract away the details in order to model these scenarios using game theory. There is one paper that is central to this work, introducing the game of FlipIt, therefore we discuss this first.

1.2 The game of FlipIt

In the game FlipIt [27] two players, aptly named the defender and attacker, fight for control of a single resource. Each player has their own button which, when pressed, gives them control over the resource, assigning them some form of benefit. Pressing the button incurs a certain cost,

which can differ for both players. The key aspects of this game are twofold. First, the players can press their respective buttons whenever they like, i.e. the game is played in continuous time. Second, the game is played in a stealthy manner. In other words, if a player presses their button, their opponent does not find out until they press their own button. The key question then becomes, when should the players press their button? The original FlipIt paper studies this in quite some detail, considering many different strategies on how to play the game, based on how much feedback the players receive throughout the game. We discuss this more formally in a later section.

FlipIt has gained traction and popularity due to the assumption that the adversary can always get into the system. The game aligns with the new rhetoric in the security industry, discussed in Section 1.1, that compromise avoidance is no longer a realistic possibility and the goal is now about limiting the amount of compromise as quickly and efficiently as possible. The original FlipIt paper [27] examines this game as a way of modelling attacks in which a stealthy adversary is trying to control a single resource. For example, it could be used to model an adversary who compromises passwords (the adversary's button press corresponds to a break of a system password), whilst the defender resets their passwords occasionally (via pressing their button in the game).

In the original FlipIt game, both the attacker and defender are 'stealthy' in the sense that neither knows if the other controls the resource before they execute a button press. The authors also explore a large number of possible strategies that the players can enforce. These are broken into various subcategories with the largest two being adaptive and non-adaptive. With a non-adaptive strategy, the players make a decision as to how they will play at the beginning of the game, and take into account any information available to them at the time, such as reward functions or costs of moving. With an adaptive strategy, the players are able to receive information regarding their opponent's play throughout the game and update their own play to accommodate this new information.

In their follow up paper [3], the authors discuss applications of the game to various examples including credential management, virtual machine refresh and cloud auditing for service-level-agreement. Our whole body of work considers different extensions of the game of FlipIt. Other authors have already considered a few, which we discuss before introducing our contribution.

1.3 Extending the game of FlipIt

Despite its simplicity, the FlipIt game is rather complex in terms of the possible different attacker and defender strategies and can be modified in various ways. Therefore, the game of FlipIt [27] has been extended in multiple directions by various authors. In this section, we discuss the various extensions in preparation for our own contributions to the area.

Laszka et al. [15] extend the FlipIt game to consider different types of attacks. Following

Herley [11], the authors split them into two categories; targeted and non-targeted. They consider the targeted attacker to be a single attacker that, once he decides to compromise the system, takes a certain time interval to successfully do so. The time interval is defined to be a random variable following an exponential distribution. If the defender moves within this time interval the attack fails. The non-targeted attacks are viewed to be multiple attackers arriving at rates following a Poisson process. The game is defined to be asymmetric, this means the attacker's moves are completely stealthy whilst being able to see all moves made by the defender. The attacker's reward function is the amount of time in control of the resource multiplied by the attacker's gain per unit of time minus the cost of moving. Thus, the game is non zero-sum. The paper argues that since the defender has the same amount of information throughout the game an adaptive strategy is no use to her. Therefore, the defender plays with a renewal strategy that follows a specific distribution, chosen at the beginning of the game. From this, since the defender does not change her strategy adaptively, Laszka et al. [15] argue that the attacker need not change either and can follow a probability distribution to choose the wait time for the attacker to move. Once the attacker has begun his move, the amount of time to compromise follows an exponential distribution at a specific rate. Therefore, a probability distribution for the time to attacker compromise can be formulated and from here the game can be fully analysed. Laszka et al. [15] find that even against an adaptive attacker the periodic strategy is the most successful strategy.

Zhang et al. [29] claim the incentives of adversaries are continually being misunderstood, and this sometimes causes more of an issue than a lack of understanding "technical mechanisms". Therefore, game theory has tried to plug this gap. Zhang et al.'s own attempt is a two-person, zero-sum game that models stealthy attacks whilst considering resource constraints. The game is played over a continuous time horizon with the cost of moving for each player varying over the multiple independent nodes. Each node can be in one of two states, compromised or protected, representing ownership by the attacker or defender, respectively. Whoever made the last move on a specific node has current control of that node. They consider an asymmetric approach, meaning the attacker and defender do not have the same "rights". Here, the attacker is considered stealthy and the defender fully observable. This means that at all times the attacker knows the full move history of the defender and can view the current state of any node, while the defender has no idea about her opponent or the state of the node. Zhang et al. [29] consider a non-uniform reward for each node, multiplying the proportion of time each player is in control of the node by that reward. Both players are assumed to be playing a non-adaptive strategy, meaning they choose a strategy at the beginning of the game and continue to use that throughout the game. The authors show that for the defender playing against a non-adaptive, independent and identically distributed (*i.i.d*) attacker, the best response for the defender is a periodic strategy. Likewise, they show that when the defender plays with a periodic strategy decided before the beginning of the game, there is no need for the attacker to be adaptive.

Pham and Cid [23] introduce a new mechanism whereby a player can test who controls the resource. This models the situation where it can be cheaper to check on the state of the resource than to take action resetting it; it can also save a wasted move. Thus a 'peek'/'probe' at the resource state costs less than taking control of the resource. One can think of this as a way of removing the stealthiness from the FlipIt game. Pham and Cid [23] then move on to discuss situations where a resource becomes hardened over time; this means that every time a player moves on a resource they already control, part of the move consists of making it harder for the opponent to regain control of the resource. This could represent a scenario where the administrator of a system might patch a weakness in a system that is brought to their attention by the adversary infiltrating the system.

The main prior work related to this thesis is that of Laszka et al [14]. This is FlipIt for multiple resources in which the attacker must compromise all resources in order to have any benefit; thus modelling an attacker against a full threshold secret sharing scheme. They consider two types of strategies defined by periodic and non-arithmetic renewal processes[3]. The authors consider both synchronised and independent types of flipping. Synchronised means when one button is pressed, all buttons are pressed. Whereas independent means the buttons can be pressed separately and at different rates. The paper establishes some basic facts on these strategies, but does not consider constructing full benefit functions for either of these strategies, nor does it find analytic Nash equilibria for the strategies. They comment on the difficulty of analytically finding Nash equilibria for the games. They state there is a simple iterative algorithm to find certain Nash equilibria, however it will not converge for the majority of cases. This is due to the analytic difficulty in obtaining such formulae. The paper moves on to considering strategies arising from Markov processes. They develop a model for two resources, considering discrete time steps and set up a linear programming solution that becomes more complicated as the finite time horizon extends.

Wellman et al. [28] also consider a number of extensions of the FlipIt paper, and much like that of Laszka et al. comment on the difficulty of obtaining analytic solutions to the Nash equilibrium. Therefore, they adopt a simulation based method. The attacker's probability of compromising increases progressively with probing, while the defender uses a moving-target technique to erase attacker progress. The paper extends the model to multiple resources and considers a time dependent 'reimage' initiated by the defender. In addition, Wellman et al. [28] set up a situation of asymmetric stealth in that the attacker can always tell when the defender has moved. The defender however, does not know when the attacker has compromised the resource, only finding out when she probes the resource.

[3]A renewal process is called non-arithmetic if there is no positive real number $d > 0$ such that the inter-arrival times are all the integer multiples of d.

1.4 Our work

In this body of work we study partial (or non-full) threshold cases, where the attacker is not required to gain control of the whole system but only a fraction of the number of resources in order to have some benefit. This is motivated by a number of potential application scenarios which we now outline:

- Large web sites usually have multiple servers responding to user requests so as to maintain high availability and response times. An APT attack on a web site may try to knock out a proportion of the servers so as to reduce the owner's quality of service below an acceptable level.

- Large networks contain multiple paths between different nodes, again to protect against attacks. An attacker will not usually be successful if he knocks out a single path, however knocking out all paths is overkill. There will be a proportion of the paths which will result in a degradation of the network connectivity which the attacker may want to achieve.

- In many computer systems multiple credentials are needed to access a main resource. Therefore, an attacker only needs to obtain enough credentials to compromise a main resource. Thus modelling attacks on credentials (e.g. passwords, certificates, etc) should really examine the case of multiple credentials in the partial threshold case.

- Multi-Party Computation (MPC) has always used threshold adversaries; an external attacker trying to compromise a system protected with MPC technology will only be interested in obtaining a threshold break above the tolerance limit of the MPC system. In such a situation, one is interested in proactively securing MPC systems, since when modelled by FlipThem a defender may regain control of a compromised party.

- Related to the last point is the case of fault tolerance. It is well known that Byzantine agreement is not possible if more than $n/3$ of the parties are compromised. Thus an adversary who simply wants to inject errors into a network protected by some Byzantine agreement protocol only needs to compromise more than $n/3$ of the servers.

Thus, we examine variants of the FlipThem game of [14] in which an attacker is trying to obtain control of at least k of the n resources. We call this the (n,k)-FlipThem game.

In Chapter 2, we formally define the multi-rate (n,k)-FlipThem, assigning independent move costs and thus move rates to each resource for each player. We present the most general framework here, and restrict certain assumptions in later chapters when required.

In Chapter 3, we introduce the game of single-rate (n,k)-FlipThem, restricting the number of rates to just one per player, whilst keeping the game played over n resources and a threshold of k. For the defender we allow two possible moves; in the first type the defender gains control of *all* resources with a single play. This models the situation where a defender might reset and

reinstall a cloud application in one go, or reset all credentials/passwords in a given move; we call this a *full reset*. In the second type of move the defender needs to select a single resource to reset. We call this type of defender move a *single reset*. We introduce continuous time Markov chains as a method of finding the benefit functions and calculating Nash equilibria of the two player partial threshold FlipThem game.

In Chapter 4, we look at the multi-rate (n,k)-FlipThem game in three strategy spaces. In each case, we build from the ground up, starting with the game of FlipIt, followed by full threshold FlipThem and finally (n,k)-FlipThem. We do this because certain analysis is only available in certain games and for certain strategy classes. The order of strategy spaces we consider are:

- We assume the players play within the periodic strategy class.
- Next, we generalise the game to the renewal strategy class.
- Finally, we consider a strategy class within the renewal space, the *exponential* class.

In the final sections of Chapter 4, we calculate the best rates for these players in response to the benefit functions and the rates of the opponent. Then the set of rates where both players are playing their best responses to each other is calculated. These are commonly known as Nash equilibria. Putative equilibria of the game are found for the benefit functions and conditions calculated to see when these equilibria are valid.

From here, we introduce two learning frameworks for the game of (n,k)-FlipThem. Adaptive frameworks such as these are required once we drop the unrealistic assumption that players know their opponent's costs and reward functions, and our version makes considerably weaker assumptions on the information available to the players than the adaptive framework of FlipIt [27].

In Chapter 5, we introduce *fictitious play*. Using results from games literature [8], we model the situation where each player responds to the observed actions of the other player over a continuing sequence of epochs, instead of calculating an equilibrium.

In Chapter 6, we go one step further by completely removing the assumption that players know analytically their payoff functions and move costs. To do this, we use *genetic algorithms* (GAs). The GA method allows us to create a population of strategies for each player and evolve the population over many generations to see if the population converges to the optimum strategy calculated in previous chapters. In each generation, strategies from the defender's population are paired up against strategies from the attacker's population in order to play the game. The only value that is returned is the specific payoff for that game. Thus, this provides a much more general framework for testing adaptive strategies against each other.

In Appendix A, we describe the FlipThem simulation lab. This is a python framework we have designed to create and test strategies within the game of threshold FlipThem. The library can be used to play individual games between varying strategies and record the results. The

games can be animated to give a picture of how strategies perform against one another. Note that all graphics displayed in this work have been created using the FlipThem simulation lab.

Tournaments between varying strategy classes can also be set up. This will randomly pick a proportion of strategies to play against each other and record performance, which gives a stronger idea of how a strategy will perform against a general population of strategies rather than one individual instance. All of this is used in our genetic algorithm implementation developed in Chapter 6, allowing any strategy class to be plugged in to assess how dominant a strategy is against an opponent population. This is a large step forward in the ability to analyse an infinite number of possible strategy classes. The library is designed using object-oriented programming (OOP) principles and therefore has scope to be developed further. We offer this as open-source software, allowing researchers to explore their own defined strategy classes.

1.5 Publications

- **Threshold FlipThem: When the winner does not need to take all**, co-authored with David Leslie and Nigel P. Smart, published at the 6th International Conference on Decision and Game Theory for Security 2015;
- **Multi-Rate Threshold FlipThem**, co-authored with David Leslie and Nigel P. Smart, published at European Symposium on Research in Computer Security 2017.

CHAPTER 2

DEFINING MULTI-RATE (n,k)-FLIPTHEM

In this chapter, we define the threshold FlipThem framework more formally. Largely, we use the notation derived from the papers [14, 27]. We choose notation from either of these works based on convenience and ability to extend to the more general framework we will be exploring. Unsurprisingly, we follow the same order and flow as [27] when defining the framework.

2.1 Formal definition

In this section, we formally introduce the game of *Threshold FlipThem*. Throughout this work we study various forms of the game; however we try to introduce everything required for subsequent sections here.

Players The game has two players - the defender and attacker. We usually consider the defender to be the "owner" of the system and denote her by D or 0 and the attacker by A or 1.

Time The game is played in continuous time, starting at $t = 0$ and running to $t \to \infty$. In previous papers introducing the game [14, 27], the authors considered both continuous and discrete time. However, we do not explore discrete time models here.

Resource state The game consists of resources $\mathscr{R}_r$ for $1 \leq r \leq n$. Each resource $\mathscr{R}_r$ has two possible states, the resource is controlled by either the defender or attacker. We define the resource state $C_r = C_r(t)$ to denote the player in control of resource $\mathscr{R}_r$. If the attacker has compromised resource $\mathscr{R}_r$ at time t we write $C_r(t) = 1$, and 0 for the defender. We note that the resource always begins in a good state, such that $C_r(0) = 0$. We also denote $Z_r^i(t)$ to be the amount

of time since Player i has moved on resource $\mathscr{R}_r$. With this, we can rewrite C_r

$$C_r(t) = I\left(Z_r^D(t) \ > \ Z_r^A(t)\right)$$

where $I()$ is the indicator function evaluating to 1 if true, and 0 otherwise.

Threshold The threshold $k \leq n$ represents the number of resources $\mathscr{R}_r$ the attacker needs to compromise in the game. To stop the attacker from doing this, the defender must be in control of at least $n-k+1$ resources. Denote the current number of resources $\mathscr{R}_r$ compromised by the attacker at time t to be $Q(t)$ so that $0 \leq Q(t) \leq n$.

Game state Define the game state $C = C(t)$ to denote the player in control of the system (i.e. above their respective threshold) as either 0 or 1. If the threshold for the attacker is k then $C(t) = 1$ at time t if and only if $Q(t) \geq k$. Therefore, we can write $C(t)$ as

$$C(t) = I(Q(t) \ \geq \ k) = I\left(\left|\left\{j \text{ s.t. } Z_j^D(t) \ > \ Z_j^A(t)\right\}\right| \geq k\right)$$

where $I()$ is the indicator function evaluating to 1 if true, and 0 otherwise. We note that the game state always begins in a good state, such that $C(0) = 0$ and $Q(0) = 0$.

Moves Since the game is played in continuous time, a player can move (press his/her button) at any time. However, as in the original FlipIt paper [27] we only allow a finite number of moves within a finite interval. We denote the move times on resource $\mathscr{R}_r$ of both players to be the infinite nondecreasing sequence

$$\mathbf{t}_r = t_{r,1}, t_{r,2}, t_{r,3}, \ldots.$$

We note that this sequence is not necessarily strictly increasing since we allow both players to move at the same time on a resource. Denote the player who made the j-th move on resource $\mathscr{R}_r$ to be $p_{r,j} \in \{0,1\}$. We define $\mathbf{p}_r$ to be the sequence representing the order of player moves on resource $\mathscr{R}_r$ such that

$$\mathbf{p}_r = p_{r,1}, p_{r,2}, p_{r,3}, \ldots.$$

Let $\mathbf{t}_r^i$ be the sequence of Player i's move times on resource $\mathscr{R}_r$ such that

$$\mathbf{t}_r^i = t_{r,1}^i, t_{r,2}^i, t_{r,3}^i, \ldots$$

Therefore, the resource $\mathscr{R}_r$'s state $C_r(t)$ represents the player that has most recently made a move on that resource such that

$$C_r(t) = p_{r,j} \text{ for } t_{r,j} < t \leq t_{r,j+1} \text{ for all } j \geq 1.$$

In other words, if $C_r(t) = 0$ then the defender has most recently moved on resource $\mathscr{R}_r$, and likewise $C_r(t) = 1$ for the attacker.

Denote $n_r^i(t)$ to be the number of moves made by Player $i \in \{D, A\}$ on resource $\mathscr{R}_r$ up to time t. Define $n_r(t)$ to be the total number of moves on resource $\mathscr{R}_r$ such that

$$n_r(t) = n_r^D(t) + n_r^A(t).$$

Denote the average move rate by Player i on resource $\mathscr{R}_r$ up to time t to be

$$\eta_r^i(t) = \frac{n_r^i(t)}{t}$$

We let $m_r^i(t)$ denote the time of the most recent move by player i on resource $\mathscr{R}_r$, this is the largest value of $t_{r,j}$ that is less than t.

Move costs For each resource $\mathscr{R}_r$, we assign move costs c_r^i for Player $i \in \{A, D\}$. We also define the ratio between players' move costs to be $\rho_r = \frac{c_r^A}{c_r^D}$. We find studying ratios between move costs can give a better understanding of player behaviours when optimising their play.

Game types The general flow of this work will consider two games already defined in the literature, named FlipIt and FlipThem. From here, we will extend these with our own game named Threshold FlipThem. FlipIt is demonstrated in Figure 2.1. The game is played over one resource, with the defender playing in blue and attacker in red. The circles represent when each player moves, and the blue and red rectangles show when the player is in control of the single resource they are playing over.

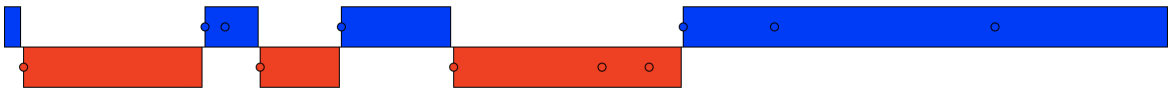


Figure 2.1: The FlipIt game played by a defender (blue) and attacker (red).

In Figure 2.2, we see Laszka et. al's [14] game of FlipThem, in which the defender and attacker are playing over multiple resources (in this case 3) and the attacker must be in control of all resources in order to achieve some gain, defined in a later section.

Finally, in Figure 2.3 we see our game of Threshold FlipThem in which the attacker need not compromise all resources in order to control the system. In this case, the attacker must control 2 of the 3 resources.

Feedback during the game One of the most important aspects of the standard version of our game Threshold FlipThem is that moves are stealthy. In later chapters we will also consider when players receive information about their opponent's moves after they have moved themselves. We consider this to be *feedback*. We define versions of ϕ_r^i to be the feedback received by Player i when they move on resource $\mathscr{R}_r$. We define two types:

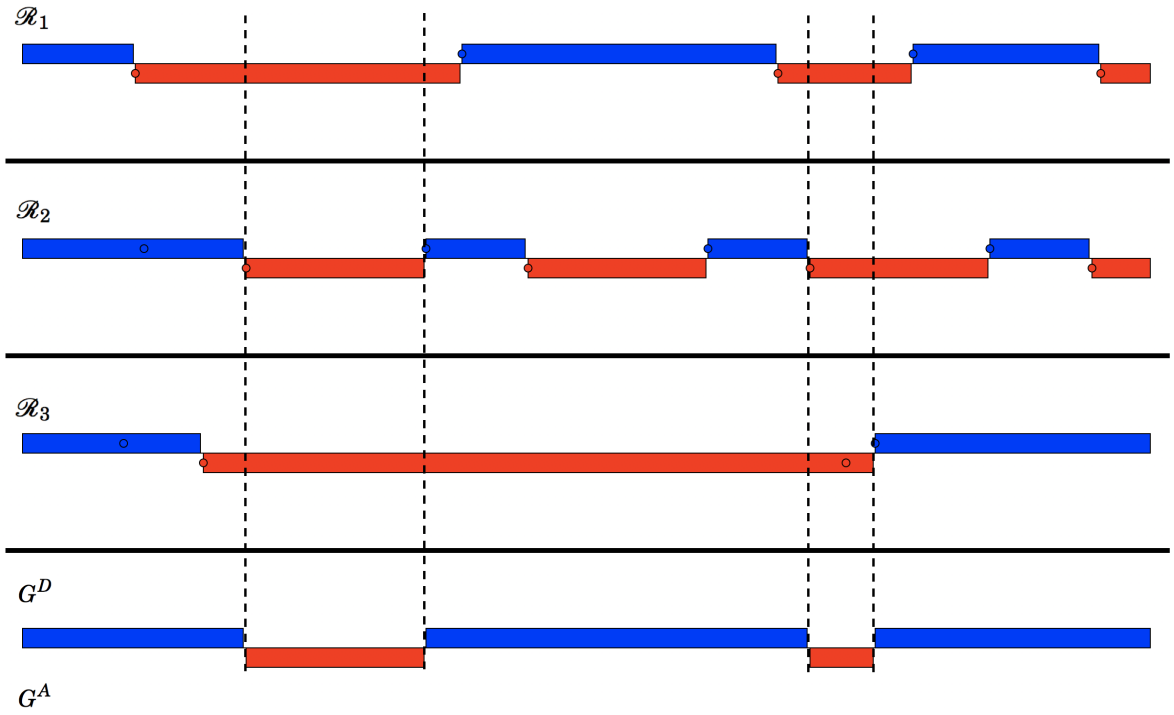


Figure 2.2: The FlipThem game played by a defender (blue) and attacker (red). Played over 3 resources and a full threshold.

- **Nonadaptive (NA):** A player does not receive any feedback when they move, meaning the feedback function is constant,

$$\phi_r^i(t_j) = 0$$

- **Last Move (LM):** The player moving at time t_j finds out the exact time their opponent last moved,

$$\phi_r^i(t_j) = m_r^{-i}(t_j)$$

where $m_r^{-i}(t_j)$ is the last move made by Player i's opponent before Player i's move. We further define the last move feedback in Chapter 6.

Views A *resource view* is a particular player's history of the resource from their specific viewpoint from the beginning of the game up to a time t. Since the player could have received information during the game, it not only lists their move times but also all information they received at each move, such that

$$\mathbf{v}_r^i(t) = ((t_{r,1}^i,\ \phi_r^i(t_{r,1}^i)),\ (t_{r,2}^i,\ \phi_r^i(t_{r,2}^i)),\ \ldots,\ (t_{r,j}^i,\ \phi_r^i(t_{r,j}^i)))$$

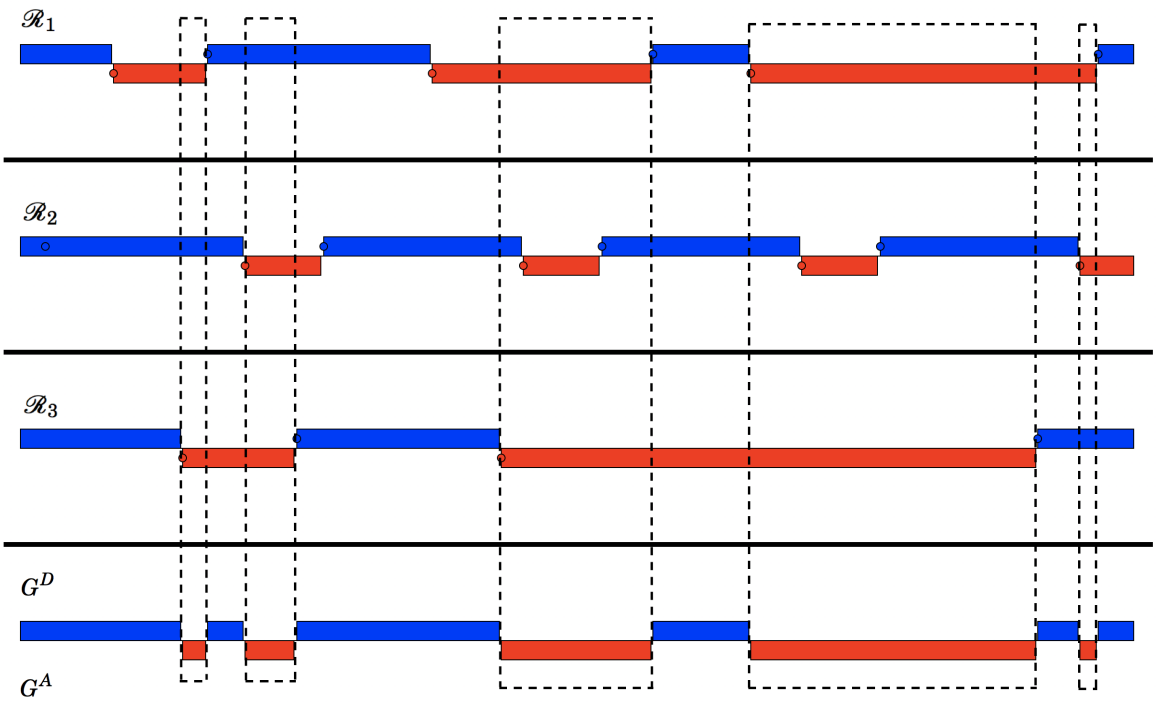


Figure 2.3: The ThresholdFlipThem game played by a defender (blue) and attacker (red). The game is played over 3 resources with the attacker threshold as 2.

where $t^i_{r,j}$ is player i's last j-th move on resource $\mathscr{R}_r$ such that $t_{r,j} < t$. We define a *view* to be the player's complete viewpoint of their moves and feedback received on every resource $\mathscr{R}_r$ such that

$$\mathbf{v}^i(t) = \left(\mathbf{v}^i_1, \mathbf{v}^i_2, \ldots, \mathbf{v}^i_n\right)^T \tag{2.1}$$

where X^T is the transpose of the matrix X.

2.1.1 Strategies

A *strategy* S^i for playing the game is a tuple of resource strategies $S^i = (S^i_1, S^i_2, \ldots, S^i_n)$ where S^i_r is player i's specific resource strategy on resource $\mathscr{R}_r$. A *resource strategy* is a mapping from a view to a positive real number, representing the next move time. Thus, if we define $\mathbf{v}^i(t)$ to be Player i's view of the game at time t as in equation (2.1) and assume Player i has moved j times on resource $\mathscr{R}_r$, then $S^i_r(\mathbf{v}^i(t))$ denotes the time for the player to wait until making their $(j+1)$-th move on resource $\mathscr{R}_r$, so that $t_{r,\,j+1} = t_{r,\,j} + S^i_r(\mathbf{v}^i(t))$. Therefore, the next resource view for Player i following strategy S^i_r on resource r would be

$$((t_{r,1},\ \phi^i_r(t_{r,1}),\ (t_{r,2},\ \phi^i_r(t_{r,2}),\ \ldots,\ (t_{r,j+1},\ \phi^i_r(t_{r,j+1})).$$

Now we start to define different strategy classes that will be explored throughout this work. We follow very similar definitions to the authors in [27] and [14].

Non-adaptive strategies: A strategy is said to be non-adaptive if it is fixed from the beginning and does not change throughout the game. No extra feedback (or views) throughout the game is acquired and therefore nothing is changed once the game has begun. In theory, a player could decide on all of their move times before the game has begun since the moves are independent of the opponent's moves. Bear in mind however that the strategy is not necessarily deterministic. It can in fact depend on a source of randomness generated from a specific probability distribution.

Periodic strategies: A periodic strategy is a non-adaptive strategy characterised by a fixed interval δ between successive moves. We assume the first move time, named the *phase*, is chosen uniformly at random within the time interval $[0,\delta]$. The average rate of play is denoted $\omega = 1/\delta$ and we define P_ω to be the periodic strategy with average rate of play ω and the class of all periodic strategies

$$\mathscr{P} = \{P_\omega | \omega > 0\}.$$

Non-Arithmetic renewal strategies: A renewal process is called non-arithmetic if there does not exist a positive real number $d > 0$ such that the inter-arrival times between events are all integer multiples of d. A non-arithmetic renewal process is generated by a probability density function f and is denoted R_f. The class of all non-arithmetic renewal strategies is denoted by $\mathscr{R}$.

Exponential strategies: An exponential strategy is a non-arithmetic renewal strategy. We define the exponential strategy E_μ with flip rate μ to have inter-arrival times generated by the exponential probability density function $f(t) = \mu e^{-\mu t}$. We define the class of all exponential strategies to be

$$\mathscr{E} = \{E_\mu | \mu > 0\}.$$

Adaptive strategies: The class of adaptive strategies characterises the strategies in which players receive feedback during the game and make decisions based on that feedback. We will define these in more depth in later chapters.

A comment on notation: In this work, the majority of the analysis will focus on general renewal strategies, the periodic strategy class and the exponential strategy class. Therefore, so the reader always knows what class we are currently working on we define the following conventional notation. When working on general renewal strategies we use α^D and α^A for the rates of the defender and attacker, respectively. For the periodic strategy class the defender's rate is denoted ω and the attacker's σ. Finally, for the exponential strategy class the defender's rate is denoted by μ and the attacker's λ. This is summarised in table 2.1.

Strategy Class	Defender	Attacker
General Renewal	α^D	α^A
Periodic	ω	σ
Exponential	μ	λ

Table 2.1: Summary of notation used for strategy classes

Flip types In this work we consider two types of flipping - *multi-rate* and *single-rate*. Multi-rate is the more general type and we have been defining it in this section. We give a brief overview in order to see the differences to single-rate. In multi-rate, for each Player i and resource $\mathscr{R}_r$ we have a cost of moving c_r^i and an associated rate of play α_r^i. Thus, these rates can be considered completely independent of each other and whilst in a lot of cases each resource rate will follow the same distribution (with different rates), in later cases we do not always make this assumption. In single-rate, Player i has just one move cost c^i and one move rate α^i. When it is time for Player i to move they will decide to move on a resource of their choice. Whichever resource Player i decides to move on will incur this cost. In most cases we assume that when it is time for the player to move they pick one at random or under some form of resource choosing strategy which we discuss in detail in later sections. Single-rate is considered in Chapter 3.

Reset types We consider two types of reset $\mathscr{R} \in \{\mathscr{F}, \mathscr{S}\}$. In the *single reset* (or individual reset) game $\mathscr{S}$, each player has a set of n buttons; there is one button on each resource $\mathscr{R}_r$ which when pressed will give control of that resource to the player. In the *full reset* game $\mathscr{F}$, for the defender there is a "master button" which simultaneously returns all resources to the defender's control. Pressing the master button costs the defender a value which we shall denote by D_n, the value of which depends on n, the number of resources. The reason for having a master button is to capture the case when resetting the entire system in one go is simpler than resetting each resource individually. For both forms of the game, the attacker has a set of n buttons, one for each resource. When the attacker presses a resource's button it will allow the adversary control of that resource, or again do nothing if the resource is already under the attacker's control.

Gains and benefits The attacker's total *gain* G^A is the amount of time he is in control of the system, so that

$$G^A(t) = \int_0^t C(t)dt = \int_0^t I(Q(t) \geq k)\,\mathrm{d}t$$

where k is the attacker's threshold value. Thus $G^i(t)$ represents the total amount of time the player has been in control of the system up to time t and so we have

$$G^A(t) + G^D(t) = t.$$

Thus, we have a formula for the defender's gain

$$G^D(t) = t - G^A(t) = t - \int_0^t I(Q(t) \geq k)\,\mathrm{d}t.$$

We want to normalise this time to improve our analytics of the game, so we define our average gain $\gamma^i(t)$ for Player i as

$$\gamma^i(t) = \frac{G^i(t)}{t}$$

Therefore, $\gamma^i(t)$ is the proportion of time Player i is in control of the system (above their required threshold) up to time t. Obviously we have for all $t > 0$

$$\gamma^D(t) + \gamma^A(t) = 1$$

We denote Player i's cost of moving on resource $\mathscr{R}_r$ by c_r^i. Define $B^i(t)$ to be Player i's *net benefit* up to time t. By this we mean the total gain up to time t minus the cost of all of Player i's moves:

$$B^i(t) = G^i(t) - \sum_{r=1}^{n} c_r^i \cdot n_r^i(t)$$

Let $\beta^i(t)$ be Player i's *average benefit* up to time t:

$$\beta^i(t) = \frac{B^i(t)}{t} = \gamma^i(t) - \sum_{r=1}^{n} c_r^i \eta_r^i(t).$$

We define Player i's *asymptotic benefit* as

$$\beta^i = \lim_{t\to\infty} \inf \beta^i(t). \tag{2.2}$$

Finally, if we represent Player i's strategy as

$$S^i = (S_1^i, S_1^i, S_2^i, \ldots, S_n^i)$$

where S_r^i is Player i's specific resource strategy on resource $\mathscr{R}_r$. The benefits of each player will correspond to the strategies chosen, and therefore we represent their benefits as $\beta^i(S^D,\ S^A)$.

2.1.2 Game theory

We denote the game of Threshold FlipThem by $(n,k)-\mathrm{FlipThem}_{\epsilon}^{\mathscr{R}}(\mathscr{C}^D,\mathscr{C}^A)$, where Player i has chosen a strategy from class $\mathscr{C}^i$ for $i \in \{A,D\}$. n is the number of resources and k the attacker threshold. ϵ is the lower bound of the defender's rate and $\mathscr{R} \in \{\mathscr{F},\mathscr{S}\}$ is the reset type. The benefits $\beta^D(S^D,S^A)$ and $\beta^A(S^D,S^A)$ are defined by (2.2) and are analogous to utilities or rewards functions in game theory. Using terminology from game theory:

- A strategy $S^D \in \mathscr{C}^D$ is *strongly dominated* for the defender D in the game $\mathrm{FlipThem}(\mathscr{C}^D,\mathscr{C}^A)$ if there exists another strategy $\widetilde{S^D}$ such that

$$\beta^D(S^D,S^A) < \beta^D(\widetilde{S^D},S^A), \quad \forall\ S^A \in C^A$$

- A strategy $S^D \in \mathscr{C}^D$ is *weakly dominated* for the defender D in the game FlipThem($\mathscr{C}^D, \mathscr{C}^A$) if there exists another strategy $\widetilde{S^D}$ such that

$$\beta^D(S^D, S^A) \leq \beta^D(\widetilde{S^D}, S^A), \quad \forall\ S^A \in C^A$$

  with at least one S^A for which the inequality is strict.

- A strategy $S^D \in \mathscr{C}^D$ is *strongly dominant* for the defender D in the game FlipThem($\mathscr{C}^D, \mathscr{C}^A$) if

$$\beta^D(S^D, S^A) > \beta^D(\widetilde{S^D}, S^A), \quad \forall\ \widetilde{S^D} \in C^D, \forall\ S^A \in C^A$$

- A strategy $S^D \in \mathscr{C}^D$ is *weakly dominant* for the defender D in the game FlipThem($\mathscr{C}^D, \mathscr{C}^A$) if

$$\beta^D(S^D, S^A) \geq \beta^D(\widetilde{S^D}, S^A), \quad \forall\ \widetilde{S^D} \in C^D, \forall\ S^A \in C^A$$

Similar definitions can be given for the attacker A. For a player, a *dominated* strategy is one such that no matter what their opponent plays there is always a better option. Therefore, it makes sense that the player will never choose to play this dominated strategy under any circumstance. Within the game theory community, there is a concept of rationality. This is an assumption that a player will never play a strategy that is strongly dominated by other strategies. A *Nash equilibrium* for the game FlipIt($\mathscr{C}^D, \mathscr{C}^A$) is a pair of strategies $(S^D, S^A) \in \mathscr{C}^D \times \mathscr{C}^A$ such that

$$\begin{aligned} \beta^D(S^D, S^A) &\geq \beta(\widetilde{S^D}, S^A), \quad \forall\ \widetilde{S^D} \in \mathscr{C}^D, \\ \beta^A(S^D, S^A) &\geq \beta(S^D, \widetilde{S^A}), \quad \forall\ \widetilde{S^A} \in \mathscr{C}^A. \end{aligned}$$

In other words, we are assuming that both players are rational, in that they are both interested in maximising their benefit functions and will choose strategies (S^D or S^A) to maximise their benefit given the behaviour of their opponent. Therefore, a pair of strategies at which each player is playing optimally (within their strategy space) against the other is called a Nash equilibrium [20]. It is a fixed point in the strategy space such that neither player can increase their benefit by unilaterally changing their strategy.

CHAPTER 3

SINGLE-RATE FLIPTHEM

In this chapter, we explore the game of single-rate threshold FlipThem. We reduce each player's strategic options to one rate of play and one move cost, lowering the dimensionality of the game and simplifying the analysis required. For the defender, we allow two possible reset types. In the first type, the defender gains control of all resources with a single play. This models the situation where a defender might reset and reinstall a whole cloud infrastructure in one go, or reset all credentials/passwords in a given move; we call this *full reset*. In the second type of move, the defender needs to select a single resource to reset. We call this type of defender move *single reset*. In all of the models we consider, we allow the attacker to compromise one resource at a time. In other words, single reset.

In the prior work of Laszka et al. [14], they use results from Dijk et al. [27] to extend the game of FlipIt to deduce the analytic gain of full-threshold FlipThem, based on rates of play. The majority of the work in this chapter is based on our paper [16]. Our main results are our calculation of analytic benefit functions and Nash equilibria for the game of partial threshold FlipThem in the case of stochastic models of play. These are models in which the players' strategies are defined by some random process. The random process defines the next time point at which the player will move (with time being considered as continuous). We introduce continuous time Markov chains as a method of finding the benefit functions and calculating Nash equilibria of the two player single-rate threshold FlipThem game.

More formally, we consider the game (n,k)-FlipThem$^{\mathscr{R}}_{\epsilon}(\mathscr{E},\mathscr{E})$ where players take strategies from the *exponential* class $\mathscr{E}$. We play the game over n resources with a threshold of k for the attacker, and consider two reset types $\mathscr{R} \in \{\mathscr{F},\mathscr{S}\}$ where $\mathscr{F}$ is full reset and $\mathscr{S}$ single reset. Finally, ϵ represents the lower bound of the defender's rate of play. Having $\epsilon > 0$ recognises the fact that the defender will never actually set the reset rate to 0. It also ensures that the benefit

functions are well defined for all valid attacker-defender strategy pairs. We will not treat the choice of our ϵ to be strategic, it will be a very small number, close to zero to represent that even when the attacker has given up (plays a rate of zero) the defender will not.

We try to answer the following question: Given the ratio of costs of play between the attacker and defender and a limit for the number of resources the defender can own, what is the best set up for the defender in order to maximise their benefit function? We discuss this further in Section 3.1.3.

3.1 Obtaining Nash equilibria in continuous time for a stochastic process

In this section, we analyse various cases of our game (n,k)-FlipThem$_c^{\mathscr{R}}(\mathscr{E},\mathscr{E})$. To explain the basic analysis techniques in a simple example we first examine the game (n,n)-FlipThem$_0^{\mathscr{F}}(\mathscr{E},\mathscr{E})$. In this game the defender can perform a full reset and the attacker is trying to compromise all n servers (i.e. the full threshold case). We also, again for initial simplicity and exposition purposes, assume that the defender could decide not to play, i.e. $\epsilon = 0$. A moment's thought will reveal in practice that such a strategy is not realistic. In the later sub-sections we remove these two simplifying assumptions and examine other cases. In particular, in Section 3.1.3 when we consider the defender performing single resets, the analysis becomes more complex.

3.1.1 Simple example, (n,n)-FlipThem$_0^{\mathscr{F}}(\mathscr{E},\mathscr{E})$: full threshold, full reset

We first consider a simple example of our framework in which the time an attacker takes to successfully compromise an individual resource follows an exponential distribution with rate λ, and the defender performs a full reset, and thus regains control of all resources, at intervals with lengths given by an exponential distribution with rate μ. An alternative description is that individual resources are compromised at the arrival times of a Poisson process with rate λ, and the state is reset at the arrival times of a Poisson process with rate μ.

In this context, we think of the attacker as being stealthy, i.e. the defender does not know how many resources are compromised when she does a full reset. A moment's thought will also reveal that in this situation it makes no difference if the defender is stealthy or not; if the defender is not stealthy then the attacker will always pick an uncompromised resource to attack, whereas if the defender is stealthy then the attacker is more likely to compromise an uncompromised resource by picking one which he knows he controlled the longest time ago. Thus an attacker simply attacks each resource in turn, given some specific ordering.

We model the number of resources compromised by the attacker at time τ as a family of random variables $X = \{X(\tau) : \tau \geq 0\}$ in the finite space $S = \{0,\ldots,n\}$. Since both the defender and attacker follow memoryless strategies (with memoryless exponential random variables determining the times between changes of state) the process X is a continuous time Markov

chain. Following the analysis of continuous time Markov chains in Grimmet et al. [9], for such a process there exists an $|S| \times |S|$ generator matrix G with entries $\{g_{ij} : i, j \in S\}$ such that

$$\Pr[X(\tau+h) = j \mid X(\tau) = i] = \begin{cases} 1 + g_{ii} \cdot h + o(h), & \text{if } j = i, \\ g_{ij} \cdot h + o(h), & \text{if } j \neq i. \end{cases} \tag{3.1}$$

The generator matrix G for continuous time Markov chains replaces the transition matrix P for discrete time Markov chains; entry g_{ij} for $i \neq j$ is the "rate" of transition from state i to state j. Summing equation (3.1) over j implies that $\sum_{j \in S} g_{ij} = 0$, so that $g_{ii} = -\sum_{j \neq i} g_{ij} \leq 0$. Basic theory [9] tells us that when the chain arrives in state i it remains there for an amount of time following an exponential$(-g_{ii})$ distribution, then jumps to state $j \neq i$ with probability $-g_{ij}/g_{ii}$.

Considering our specific example with the defender using full reset, we can consider our model as a "birth-reset process" (by analogy with a "birth–death process") in which

$$\Pr[X(\tau+h) = j \mid X(\tau) = i] = \begin{cases} \lambda \cdot h + o(h), & \text{if } j = i+1, \\ \mu \cdot h + o(h), & \text{if } j = 0, \\ 1 - (\lambda+\mu) \cdot h + o(h), & \text{if } j = i, \\ o(h), & \text{otherwise.} \end{cases}$$

Thus, $g_{i0} = \mu$, $g_{i,i+1} = \lambda$, $g_{ii} = -(\mu+\lambda)$ and $g_{ij} = 0$ otherwise. From this the generator matrix can be constructed:

$$G = \begin{pmatrix} -\lambda & \lambda & 0 & 0 & \dots & 0 & 0 \\ \mu & -(\mu+\lambda) & \lambda & 0 & \dots & 0 & 0 \\ \mu & 0 & -(\mu+\lambda) & \lambda & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mu & 0 & 0 & 0 & \dots & -(\mu+\lambda) & \lambda \\ \mu & 0 & 0 & 0 & \dots & 0 & -\mu \end{pmatrix}.$$

Thus, when the state is $i \in \{1, \dots, n-1\}$ the system will jump to either state $i+1$ with probability $\lambda/(\lambda+\mu)$ (when the attacker compromises another resource before reset occurs) or to state 0 with probability $\mu/(\lambda+\mu)$ (when the reset occurs before another resource is compromised). Clearly the chain is never going to settle in one state; it will continue to randomly fluctuate between various states depending on the rates of play μ and λ. However further theory [9] indicates that the long run proportion of time the system spends in each state is given by the stationary distribution, a row vector $\pi = (\pi_0, \dots, \pi_n)$ such that $\pi G = 0$ and $\sum_{i=0}^{n} \pi_i = 1$.

Using our specific generator matrix G it can be shown that

$$\pi = \left(\frac{\mu}{\mu+\lambda}, \frac{\mu \cdot \lambda}{(\mu+\lambda)^2}, \dots, \frac{\mu \cdot \lambda^{n-1}}{(\mu+\lambda)^n}, \frac{\lambda^n}{(\mu+\lambda)^n} \right). \tag{3.2}$$

This tells us the proportion of time spent in each state. We therefore obtain the benefit functions of

$$\beta_D(\mu, \lambda) = 1 - \pi_n - c^D \cdot \mu = 1 - \frac{\lambda^n}{(\mu+\lambda)^n} - c^D \cdot \mu$$

and

$$\beta_A(\mu,\lambda) = \pi_n - c^A \cdot \lambda = \frac{\lambda^n}{(\mu+\lambda)^n} - c^A \cdot \lambda,$$

where β_D is the benefit function of the defender and β_A is the benefit function of the attacker.

Recall that in this model, when the defender plays she is resetting all resources at once. Therefore, the cost of the defender's move c^D is likely to depend on n, the number of resources. We represent this by setting $c^D = D_n$.

Using the stationary distribution described above, the benefit functions for the normalized game are

$$\beta_D(\mu,\lambda) = 1 - \frac{\lambda^n}{(\mu+\lambda)^n} - D_n \cdot \mu \quad \text{and} \quad \beta_A(\mu,\lambda) = \frac{\lambda^n}{(\mu+\lambda)^n} - c^A \cdot \lambda. \tag{3.3}$$

We are assuming that both players are rational, in that they are both interested in maximising their benefit functions, and will therefore choose a rate (λ or μ) to maximise their benefit given the behaviour of their opponent. A pair of rates at which each player is playing optimally against the other is called a Nash equilibrium [20]. At such a point neither player can increase their benefit by changing their rate; we are looking for pairs (λ^*, μ^*) such that

$$\beta_D(\mu^*,\lambda^*) = \max_{\mu \in R_+} \beta_D(\mu,\lambda^*) \quad \text{and} \quad \beta_A(\mu^*,\lambda^*) = \max_{\lambda \in R_+} \beta_A(\mu^*,\lambda).$$

Note that $\mu^* = \lambda^* = 0$ is not an equilibrium of the game defined by equations in (3.3) but is a fixed point. This is because if a player chooses to play at the zero-rate their opponent is able to move unilaterally to increase their payoff. In later sections we will bound μ below to remove this solution and for now we will search for non-trivial solutions.

Differentiating the defender's benefit function β_D with respect to μ and solving for μ gives at most one non-negative real solution, given by

$$\hat{\mu}(\lambda) = \sqrt[n+1]{\frac{n\lambda^n}{D_n}} - \lambda$$

If $\lambda < \frac{n}{D_n}$ then this is positive, and checking the second derivative confirms this corresponds to a maximum. If $\lambda \geq \frac{n}{D_n}$ then $\frac{\partial \beta_D}{\partial \mu} < 0$ for all $\mu \geq 0$ and so the optimal rate for the defender is $\mu = 0$. Hence the best response of the defender is given by

$$\hat{\mu}(\lambda) = \begin{cases} \sqrt[n+1]{\frac{n\lambda^n}{D_n}} - \lambda & \text{if } \lambda < \frac{n}{D_n} \\ 0 & \text{if } \lambda \geq \frac{n}{D_n}. \end{cases}$$

We now calculate

$$\frac{\partial \beta_A}{\partial \lambda} = \frac{n \cdot \mu \cdot \lambda^{n-1}}{(\mu+\lambda)^{n+1}} - c^A.$$

A closed form solution for λ which equates this to 0 is not easy to calculate directly. However, plugging in $\hat{\mu}(\lambda^*)$ we see that λ^* must be either 0 or satisfy

$$\frac{n \cdot \hat{\mu}(\lambda^*) \cdot (\lambda^*)^{n-1}}{(\hat{\mu}(\lambda^*) + \lambda^*)^{n+1}} - c^A = 0. \tag{3.4}$$

If it were the case that $\lambda^* \geq \frac{n}{D_n}$ then $\hat{\mu}(\lambda^*) = 0$ and there are no solutions to this equation. Note that this indicates that no equilibrium exists when the attacker's rate is too high — the intuition for this is if the attacker's rate is sufficiently high, the defender ceases to defend, and thus the attacker can do just as well by reducing their rate. Thus at any equilibrium we must have $\lambda^* < \frac{n}{D_n}$, and therefore $\mu^* = \hat{\mu}(\lambda^*) = \sqrt[n+1]{\frac{n(\lambda^*)^n}{D_n}}$. Plugging this back into equation (3.4) we see that either

$$\lambda^* = \frac{n \cdot D_n^n}{(D_n + c^A)^{n+1}}, \quad \mu^* = \hat{\mu}(\lambda^*) = \frac{n \cdot c^A \cdot D_n^{n-1}}{(D_n + c^A)^{n+1}}, \tag{3.5}$$

or $\mu^* = \lambda^* = 0$. The non-zero solution will only correspond to a Nash equilibrium if $\beta_A(\mu^*, \lambda^*) \geq \beta_A(\mu^*, 0) = 0$, since otherwise λ^* is not a best response against μ^*. Note that this is the case if

$$0 < \frac{(\lambda^*)^n}{(\mu^* + \lambda^*)^n} - c^A \cdot \lambda^* = \frac{(D_n)^n}{(D_n + c^A)^{n+1}}(D_n + c^A \cdot (1-n))$$

i.e. if $c^A/D_n < 1/(n-1)$.

In the game (n,n)-FlipThem$_0^{\mathscr{F}}(\mathscr{E},\mathscr{E})$ we have defined ρ to be the ratio between the attacker and defender's costs, so that $\rho = a/D_n$. Therefore, the game Nash equilibria (NE) of the game (n,n)-FlipThem$_0^{\mathscr{F}}(\mathscr{E},\mathscr{E})$ is the empty list {} for all $\rho > 1/(n-1)$. If $\rho < 1/(n-1)$ we have a further equilibrium (μ^*, λ^*) such that the NE are the list of $\{(\mu^*, \lambda^*)\}$ where

$$\mu^* = \frac{n \cdot \rho}{D_n \cdot (1+\rho)^{n+1}}, \quad \lambda^* = \frac{n}{D_n \cdot (1+\rho)^{n+1}} = \mu^*/\rho.$$

The attacker's cost per move is independent of n, which implies that the defender will be successful, assuming $\frac{D_n}{n-1}$ is a decreasing function of n, as long as n is large enough. Thus, for the defender to always win we require the cost of a full reset to be a sublinear function of the number of resources.

In the case of resetting a cloud application this might be a reasonable assumption, but in the case of requiring n users to reset their passwords it is likely that the cost is a superlinear function as opposed to sublinear due to the social cost in needing to implement such a password policy.

3.1.2 (n,k)-FlipThem$_c^{\mathscr{F}}(\mathscr{E},\mathscr{E})$: (n,k)-Threshold, Full Reset

We now generalize the previous easy case to the threshold case (n,k)-FlipThem$_c^{\mathscr{F}}(\mathscr{E},\mathscr{E})$, i.e. we treat the number of servers which the attacker has to compromise as a parameter k, and in addition we bound the defender's strategy away from zero. Thus, the defender not playing at all is not considered a valid strategy[1]. Much of the prior analysis carries through, since we are still assuming the defender performs a full reset on her turn. Therefore, the stationary distribution is once more,

$$\pi = \Big(\frac{\mu}{\mu+\lambda}, \ldots, \frac{\mu \cdot \lambda^{j-1}}{(\mu+\lambda)^j}, \ldots, \frac{\mu \cdot \lambda^{n-1}}{(\mu+\lambda)^n}, \frac{\lambda^n}{(\mu+\lambda)^n}\Big).$$

[1] Of course, if the attacker decides not to play that is considered a good thing.

The benefit functions are now derived from the ratio of times which the attacker has compromised at least k resources, which simplifies due to the formula for geometric series:

$$\begin{aligned}\beta_D(\mu,\lambda) &= 1 - \frac{\lambda^n}{(\mu+\lambda)^n} - \sum_{i=k}^{n-1} \frac{\mu\cdot\lambda^i}{(\mu+\lambda)^{i+1}} - D_n\cdot\mu \\ &= 1 - \frac{\lambda^k}{(\mu+\lambda)^k} - D_n\cdot\mu.\end{aligned}$$

Using the same analysis, the attacker's benefit is $\beta_A(\mu,\lambda) = \frac{\lambda^k}{(\mu+\lambda)^k} - c^A\cdot\lambda$. Note that these benefit functions are identical to those in the full threshold case of the previous section, but with n replaced by k. If we were still considering the lower bound for the defender's rate of play ϵ to be zero the conclusions would be as before, but with the modification that we use k instead of n. Since we are now considering the more realistic assumption that $\epsilon > 0$ the analysis gets slightly more involved, but remains similar to that above. In particular

$$\beta_D(\mu,\lambda) = 1 - \left(\frac{\lambda}{\lambda+\mu}\right)^k - D_n\cdot\mu, \quad \text{and} \quad \frac{\partial\beta_D}{\partial\mu} = \frac{k\cdot\lambda^k}{(\lambda+\mu)^{k+1}} - D_n.$$

This derivative is decreasing in μ, and 0 at $\lambda\cdot\left[\left(\frac{k}{\lambda\cdot D_n}\right)^{\frac{1}{k+1}} - 1\right]$. It follows immediately that β_D is a unimodal function of μ, so that the maximising μ value in $[\epsilon,\infty)$ is given by

$$\hat{\mu}(\lambda) = \min\left\{\epsilon, \lambda\cdot\left[\left(\frac{k}{\lambda\cdot D_n}\right)^{\frac{1}{k+1}} - 1\right]\right\}. \tag{3.6}$$

As above, we have that

$$\beta_A(\mu,\lambda) = \left(\frac{\lambda}{\mu+\lambda}\right)^k - c^A\cdot\lambda \quad \text{and} \quad \frac{\partial\beta_A}{\partial\lambda} = \frac{k\cdot\mu\cdot\lambda^{k-1}}{(\lambda+\mu)^{k+1}} - c^A. \tag{3.7}$$

Thus for a particular value of μ the maximising λ must either be 0 or be a root of the derivative. However, explicitly solving for λ does not appear to be possible, but we note that

$$\frac{\partial^2\beta_A}{\partial\lambda^2} = \frac{k\cdot\mu\cdot\lambda^{k-2}}{(\lambda+\mu)^{k+2}}\cdot\left[\mu\cdot(k-1) - 2\cdot\lambda\right]$$

so that the first derivative, $\frac{\partial\beta_A}{\partial\lambda}$, is increasing when $\lambda < \mu\cdot(k-1)/2$ then decreasing. Since $\frac{\partial\beta_A}{\partial\lambda}$ is equal to $-c^A$ when $\lambda = 0$ and asymptotes to $-c^A$ as $\lambda\to\infty$ we have the derivative increasing from $-c^A$ to a maximum when $\lambda = \mu\cdot(k-1)/2$ then decreasing back to $-c^A$. The maximal value of $\frac{\partial\beta_A}{\partial\lambda}$ is given by

$$\frac{4\cdot k\cdot(k-1)^{k-1}}{\mu\cdot(k+1)^{k+1}} - c^A, \tag{3.8}$$

which is positive only if μ is sufficiently small. As a function of λ, β_A therefore initially decreases (from 0), has a period of increase only if μ is sufficiently small, then decreases again. It follows that β_A has at most one non-zero maximum, which occurs in the region $(\mu\cdot(k-1)/2,\infty)$ once

the derivative is decreasing, and this fixed point maximises $\beta_A(\mu,\lambda)$ on $\lambda \in [0,\infty)$ if and only if $\beta_A(\mu,\lambda) > 0$; otherwise the best response must be $\lambda = 0$. We use these insights to explore Nash equilibria directly. First consider the existence of a Nash equilibrium (μ^*,λ^*) with $\mu^* > \epsilon$. Note that if λ^* were equal to 0 then this would force $\mu^* = \epsilon$, so it must be the case that $\mu^* = \hat{\mu}(\lambda^*)$ and $\frac{\partial \beta_A}{\partial \lambda}(\mu^*,\lambda^*) = 0$. It follows from (3.6) and (3.7) that

$$c^A = \frac{k \cdot \mu^* \cdot \lambda^{*k-1}}{(\lambda^* + \mu^*)^{k+1}} = D_n \cdot \left[\left(\frac{k}{\lambda^* \cdot D_n} \right)^{\frac{1}{k+1}} - 1 \right]$$

and hence

$$\lambda^* = \frac{k}{D_n \cdot (1+\rho)^{k+1}}, \qquad \mu^* = \frac{k \cdot \rho}{D_n \cdot (1+\rho)^{k+1}}. \tag{3.9}$$

We have checked necessary conditions so far, but have still not verified that this λ^* does correspond to a maximum of β_A. As observed above, the necessary and sufficient condition is that

$$0 < \beta_A(\mu^*,\lambda^*) = \frac{1 + \rho - \rho \cdot k}{(1+\rho)^{k+1}}.$$

Thus an equilibrium of this form exists when

$$\rho < \frac{1}{k-1} \quad \text{and} \quad \mu^* = \frac{k \cdot \rho}{D_n \cdot (1+\rho)^{k+1}} > \epsilon.$$

Therefore, if the ratio ρ of the attacker's cost and defender's cost is less than $\frac{1}{k-1}$ then the game $\text{FlipThem}^{\mathscr{F}}_{\epsilon}(n,k,d,\rho)$ returns the list consisting of two pairs, the trivial equilibrium of no play (from the attacker, the defender plays at minimal rate ϵ) and an equilibrium at

$$\mu^* = \frac{k \cdot \rho}{D_n \cdot (1+\rho)^{k+1}}, \quad \lambda^* = \frac{k}{D_n \cdot (1+\rho)^{k+1}} = \mu^*/\rho.$$

Exploring the equilibria Note that if the maximal value of the derivative of β_A is non-positive then no stationary point of β_A exists, and so λ will be 0. By removing all local maxima of the attacker's payoff function we really would expect the attacker to just stop playing; i.e. this would be the perfect defenders strategy. From (3.8) we see that by taking

$$\epsilon \geq \frac{4 \cdot k \cdot (k-1)^{k-1}}{c^A \cdot (k+1)^{k+1}} \tag{3.10}$$

we can ensure there is only the trivial equilibrium. Note that a simpler lower bound on ϵ, which trivially implies the one above, is to take $\epsilon \geq \frac{4}{c^A \cdot (k+1)}$. Note that choosing a sufficiently high ϵ in this way is very conservative. The rate of decrease of β_A is $-c^A$ at $\lambda = 0$ and as $\lambda \to \infty$, so by insisting there is no local maximum at all we ensure β_A stays well away from 0.

Picking ϵ to force out the attacker only makes sense if the defender's benefit is actually maximised. It might be the case that stopping the attacker completely is not economically viable.

Therefore, in such a case ϵ should be chosen to be very small, close to zero and the other equilibria in equation (3.9) should be used; implying that μ^* is less than the right hand side of equation (3.10). Thus an expected amount of attacker success may be tolerated if completely eliminating such success comes at too much of a price. Recall our function $\mathrm{Opt}^{\mathscr{F}}_{N,\mathscr{T},\epsilon}(d,\rho)$. If we fix $\epsilon = 0.01/d$ and set $\mathscr{T} = \{k \leq n\}$, and run this programmatically for ρ from 0 to 1, Fig. 3.1 shows the smallest $n \leq N$ that maximises the defenders benefit for various N. Recall that the attacker will not play if $\rho > 1/k - 1$, meaning that as ρ increases the level of threshold decreases and therefore the number of servers required decrease. The optimum defender's benefit occurs when $k = n$. This explains the step down in Fig. 3.1.

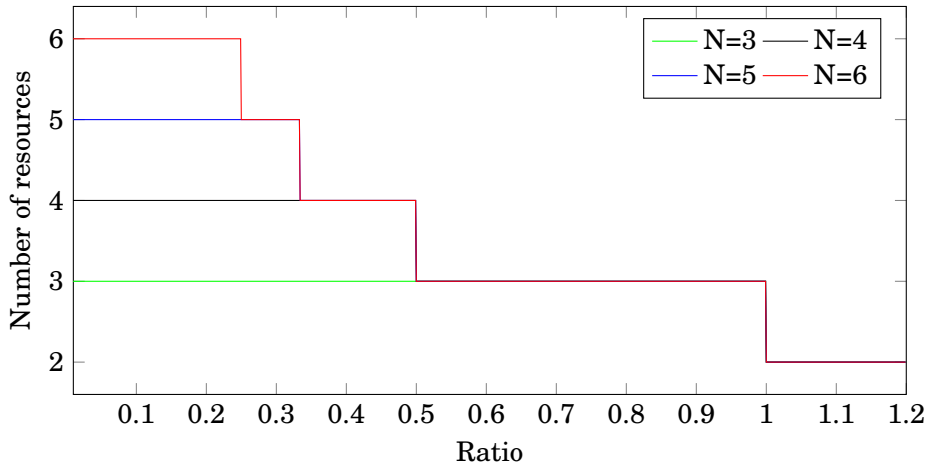


Figure 3.1: Number of resources used by the defender to maximise his benefit given a specific ρ

We end this section by examining the classic case of a threshold situation in which the required threshold is a constant fraction of the total number of resources. Suppose we have $k = \gamma \cdot n$ for some constant $\gamma \in (0,1]$. We have shown that the attacker will not play if $\frac{c^A \cdot \rho}{D_n} \geq \frac{1}{k-1} = \frac{1}{\gamma \cdot n - 1}$. As expected we see that if the attacker needs to compromise fewer resources, then the attacker's cost per resource needs to be greater for them not to play. It is intuitively obvious that the smaller the threshold the more likely the attacker will play (and succeed).

3.1.3 (n,k)-FlipThem$^{\mathscr{S}}_{\epsilon}(\mathscr{E},\mathscr{E})$: (n,k)-Threshold, Single Reset

So far, we have set up the model such that the defender can reset the whole system regaining full control whereas the attacker compromises each resource individually. We now consider the game (n,k)-FlipThem$^{\mathscr{S}}_{\epsilon}(\mathscr{E},\mathscr{E})$. The defender can reset a single machine at any specific time. Consider the situation at any time point where the number of resources compromised is j out of n. Assume the defender is going to reset a resource. There are multiple strategies they could employ, they could pick a resource which they have not reset recently, or pick a random resource, or pick a resource in a given secret sequence. Here we will assume the players pick resources uniformly at

random. Thus the probability of resetting a compromised resource is $\frac{j}{n}$, and that of wastefully resetting a non-compromised resource $1-\frac{j}{n}$. Letting the defender's and attacker's rate of play be μ and λ respectively, it is not hard to see that our generating matrix now becomes

$$G = \begin{pmatrix} -\lambda & \lambda & 0 & 0 & \dots & 0 & 0 & 0 \\ \frac{\mu}{n} & -\frac{(\mu+(n-1)\cdot\lambda)}{n} & \frac{(n-1)\cdot\lambda}{n} & 0 & \dots & 0 & 0 & 0 \\ 0 & \frac{2\cdot\mu}{n} & -\frac{(2\cdot\mu+(n-2)\cdot\lambda)}{n} & \frac{(n-2)\cdot\lambda}{n} & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \frac{(n-1)\cdot\mu}{n} & -\frac{((n-1)\cdot\mu+\lambda)}{n} & \frac{\lambda}{n} \\ 0 & 0 & 0 & 0 & \dots & 0 & \mu & -\mu \end{pmatrix}$$

We then solve for the stationary distribution $\pi = (\pi_0, \pi_1, \dots, \pi_{n-1}, \pi_n)$, by solving $\pi G = 0$, and it can be shown by induction that

$$\pi_j = \frac{n! \cdot \lambda^j \cdot \pi_0}{(n-j)! \cdot j! \cdot \mu^j} = \frac{\binom{n}{j} \cdot \lambda^j \cdot \pi_0}{\mu^j}.$$

Recall, that we also need to utilize the constraint $\sum_{i=0}^{n} \pi_i = 1$, which implies that we have $\pi_0 = \frac{\mu^n}{(\mu+\lambda)^n}$ so that we obtain the stationary distribution

$$\pi = \frac{1}{(\mu+\lambda)^n}\left(\mu^n, n\cdot\lambda\cdot\mu^{n-1}, \dots, \binom{n}{j}\cdot\mu^{n-j}\cdot\lambda^j, \dots, n\cdot\mu\cdot\lambda^{n-1}, \lambda^n\right).$$

Once again, this gives us the proportion of time spent in each state. We assume here that the costs and benefits have already been normalised and do not depend on n the number of resources. Constructing these benefit functions gives

$$\beta_D(\mu,\lambda) = 1 - \sum_{i=k}^{n} \pi_i - c^D \cdot \mu = 1 - \frac{1}{(\mu+\lambda)^n} \cdot \sum_{i=k}^{n} \binom{n}{i} \cdot \mu^{n-i} \cdot \lambda^i - c^D \cdot \mu,$$

$$\beta_A(\mu,\lambda) = \sum_{i=k}^{n} \pi_i - c^A \cdot \lambda = \frac{1}{(\mu+\lambda)^n} \cdot \sum_{i=k}^{n} \binom{n}{i} \cdot \mu^{n-i} \cdot \lambda^i - c^A \cdot \lambda$$

We want to find the Nash Equilibria for these benefit functions. A point at which neither player can increase their benefit by changing their rate. We want to find pairs (μ^*, λ^*) such that

$$\beta_D(\mu^*,\lambda^*) = \max_{\mu\in(\epsilon,\infty)} \beta_D(\mu,\lambda^*) \quad \text{and} \quad \beta_A(\mu^*,\lambda^*) = \max_{\lambda\in R_+} \beta_A(\mu^*,\lambda),$$

where ϵ is the lowest rate we can expect the defender to play in order to ensure the stationary distributions and hence benefit functions are well defined for all valid (μ,λ). Differentiating the defender's and attacker's functions with respect to μ and λ respectively gives,

$$\frac{\partial \beta_D}{\partial \mu} = \frac{n! \cdot \mu^{n-k} \cdot \lambda^k}{(k-1)! \cdot (n-k)! \cdot (\mu+\lambda)^{n+1}} - c^D, \tag{3.11}$$

$$\frac{\partial \beta_A}{\partial \lambda} = \frac{n! \cdot \mu^{n-k+1} \cdot \lambda^{k-1}}{(k-1)! \cdot (n-k)! \cdot (\mu+\lambda)^{n+1}} - c^A. \tag{3.12}$$

Closed form solutions for μ and λ which equate to 0 are not easy to calculate directly. The second derivative of the attacker's benefit with respect to λ is

$$\frac{n! \cdot \mu^{n-k+1} \cdot \lambda^{k-2}}{(k-1)! \cdot (n-k)! \cdot (\mu+\lambda)^{n+2}} \cdot [\mu \cdot (k-1) - \lambda \cdot (n+2-k)].$$

Thus, $\frac{\partial \beta_A}{\partial \lambda}$ is increasing when

$$\lambda < \frac{\mu \cdot (k-1)}{n+2-k},$$

then decreasing. Since $\frac{\partial \beta_A}{\partial \lambda}$ is $-c^A$ at $\lambda = 0$ and asymptotes to $-c^A$ as $\lambda \to \infty$ we have the derivative increasing from $-c^A$ to a maximum when $\lambda = \frac{\mu \cdot (k-1)}{n+2-k}$ and then decreasing back to $-c^A$. The maximal value of $\frac{\partial \beta_A}{\partial \lambda}$ is given by

$$\frac{n! \cdot (k-1)^{k-1}}{k^{n+1} \cdot (n+2-k)^{k-2} \cdot \mu} - c^A \tag{3.13}$$

which is positive only if μ is sufficiently small. As a function of λ, β_A therefore initially decreases (from 0), has a period of increase only if μ is sufficiently small, then decreases again. It follows that β_A has at most one non-zero maximum which occurs in the region

$$\left(\frac{\mu(k-1)}{n+2-k}, \infty\right)$$

once the derivative is decreasing, and this fixed point maximises $\beta_A(\mu, \lambda)$ on $\lambda \in [0, \infty)$ if and only if $\beta_A(\mu, \lambda) > 0$; otherwise the best response must be $\lambda = 0$. First, like the full reset case, we consider the existence of a Nash Equilibrium (μ, λ) with $\mu > \epsilon$. Since both derivatives (3.11) and (3.12) are hard to solve analytically for general n, we used a numerical method utilizing the Maple algebra system to solve for a specific n. The method for solving starts with defining the benefit functions in terms of μ and λ, we then differentiate the derivatives as above and solve for μ and λ for the defender and attacker, respectively. This provides 2 generic solutions of the form

$$\hat{\mu}(\lambda) = \text{RootOf}(f(\lambda)) \quad \text{and} \quad \hat{\lambda}(\mu) = \text{RootOf}(g(\mu))$$

where f and g are polynomials. We then put these solutions back into the derivatives to give

$$\frac{\partial \beta_D(\mu, \hat{\lambda}(\mu))}{\partial \mu} \quad \text{and} \quad \frac{\partial \beta_A(\hat{\mu}(\lambda), \lambda)}{\partial \lambda}$$

Solving these with respect to μ and λ respectively gives solutions for μ^* and λ^* with respect to the costs c^D and c^A. From this we can consider the ratio $\rho = \frac{c^A}{c^D}$ between the attacker's and defender's costs of play. A table can be constructed to show the ratios at which both the defender and attacker will and won't play for various ρ. Recall that even if the attacker is not playing, the defender must still play at some rate ϵ in order to ensure control of the system. In order to calculate the defender's benefit given a specific ρ we must calculate the lowest rate of play for the defender when the attacker is not playing. From equation (3.13), $\frac{\partial \beta_A}{\partial \lambda}$ is never positive if

$$\mu > \frac{n! \cdot (k-1)^{k-1}}{k^{n+1} \cdot (n+2-k)^{k-2} \cdot c^A}$$

Meaning no stationary point exists for the attackers benefit. From this we can see that by taking

$$\epsilon \geq \frac{n! \cdot (k-1)^{k-1}}{k^{n+1} \cdot (n+2-k)^{k-2} \cdot c^A}$$

we can ensure there is no equilibrium with $\mu^* = \epsilon$ and $\lambda \neq 0$. Recall that $\rho = \frac{c^A}{c^D}$, so that

$$\epsilon \geq \frac{n! \cdot (k-1)^{k-1}}{k^{n+1} \cdot (n+2-k)^{k-2} \cdot \rho \cdot c^D}$$

This shows that if ρ is large enough, ϵ will be small meaning the likely strategy for the attacker will be no play, $\lambda = 0$. So the benefit for the defender will be

$$\beta_D(\epsilon, 0) = 1 - \epsilon \cdot c^D = 1 - \frac{n! \cdot (k-1)^{k-1}}{k^{n+1} \cdot (n+2-k)^{k-2} \cdot \rho}.$$

However, having ρ large enough to ensure ϵ is small enough is an unrealistic assumption and choosing ϵ like this becomes a strategic choice. As it was for the full reset case, it is also very conservative and could be expensive for the defender. We therefore fix our $\epsilon > 0$ to be very small, close to zero before the game.

We now want to ask the following question: given the costs of play for both defender and attacker and a limit N for the number of resources the defender can own, what is the best set up for the defender in order to maximise their benefit function? Mathematically speaking, given ρ and N we are looking for the pairs such that

$$\beta_D^*(n^*, k^*) = \max_{n \leq N, k \leq n} \beta_D^*(n, k)$$

where $\beta_D^*(n,k) = \beta_D(\mu^*, \lambda^*)$ is the Nash equilibrium for the specific number of resources n and threshold k. We define this game to be the function $\text{Opt}^{\mathscr{S}}_{N,\epsilon}(c^D, \mathscr{T}, \rho)$. The function plays the first game (n,k)-$\text{FlipThem}^{\mathscr{R}}_{\epsilon}(\mathscr{E}, \mathscr{E})$ for all n and all k subject to some constraint space $\mathscr{T}$ [2]. The function $\text{Opt}^{\mathscr{R}}_{N,\epsilon}(c^D, \mathscr{T}, \rho)$ then finds the values of n and k which produce the greatest possible benefit for the defender. We turn to the method of numerical programming for this problem. Obviously, since the lowest rate of play ϵ for the defender is chosen arbitrarily before the game is played, if the equilibrium played is the trivial equilibrium then the defenders benefit is $\beta_D(\epsilon, 0) = 1 - \epsilon \cdot c^D$.

When running $\text{Opt}^{\mathscr{S}}_{N,\epsilon}(c^D, \mathscr{T}, \rho)$, each round of (n,k)-$\text{FlipThem}^{\mathscr{S}}_{\epsilon}(\mathscr{E}, \mathscr{E})$ played has three possible outcomes.

- If ρ is so small the defender will play at the minimal possible rate ϵ.
- If ρ is "mid-size" the defender and attacker both play the non-trivial equilibrium (μ^*, λ^*).
- If ρ is large the attacker does not play and the trivial equilibrium $(\epsilon, 0)$ is played.

[2] For example $k \leq n$, or $k \leq n/2$, or $n - t \geq B$ for some bound B.

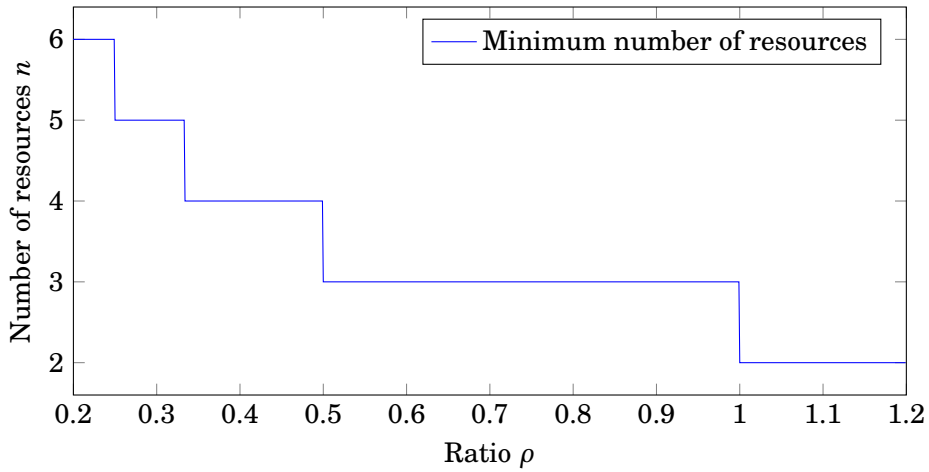


Figure 3.2: Number of resources used by the defender to maximise her benefit given a specific ρ, for $\mathcal{T} = \{k \leq n\}$ and $N = 7$.

We experimentally examined two scenarios, both in which we fix $\epsilon = 0.01/c^D$. In the first scenario we take $\mathcal{T} = \{k \leq n\}$ and $N = 7$, in this case the function $\mathrm{Opt}^{\mathscr{S}}_{N,\epsilon}(c^D, \mathcal{T}, \rho)$ outputs valid configurations for relatively small values of ρ, see Fig. 3.2. Interestingly, the setup that outputs the maximum defender's benefit is always a full threshold game. Also, as the ratio between move costs increases, the required number of resources decreases.

In the second scenario, we take $\mathscr{T} = \{k \leq n/2\}$, and again $N = 7$. The results are given in Fig. 3.3. In this case, small values of ρ result in games for which the defender will not play, for larger values of ρ we end up requiring more servers.

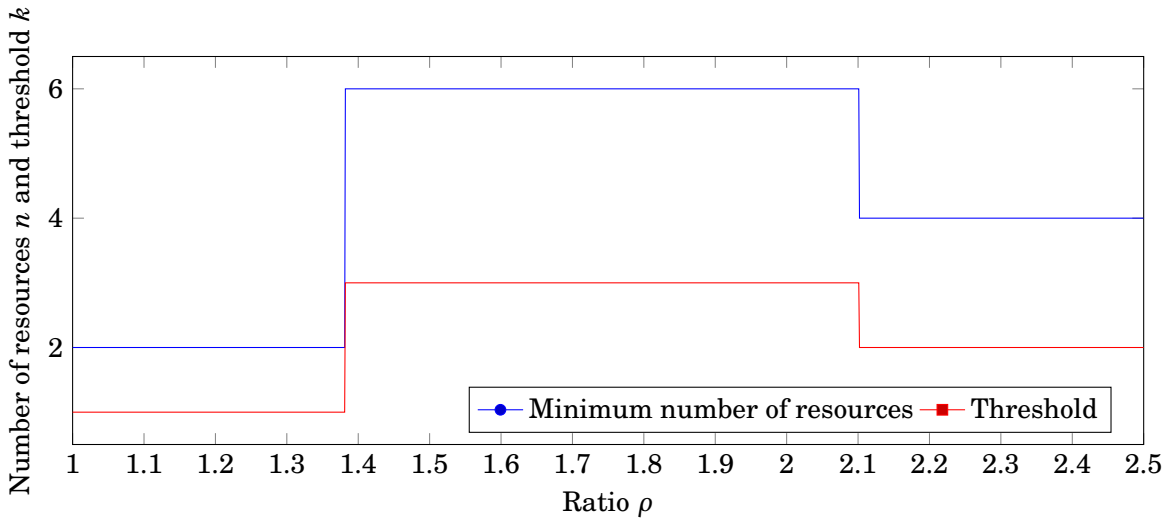


Figure 3.3: Number of resources used by the defender to maximise her benefit given a specific ρ, for $\mathscr{T} = \{k \leq n/2\}$ and $N = 7$.

3.2 Summary

In this chapter, we analysed various cases of our single-rate (n,k)-FlipThem$^{\mathscr{R}}_{\mathscr{C}}(\mathscr{E},\mathscr{E})$ game. For full reset, we found that the equilibria depend solely on the threshold of the resources and the costs of play, not the number of resources involved. As the cost for the attacker increases, the necessary amount of servers (threshold) required for the defender to maximise her benefit decreases.

For single reset, the analysis is harder by hand. We asked the question: "Given the costs of play for both defender and attacker and a limit for the number of resources the defender can own, what is the best set up for the defender in order to maximise her benefit?" Based on the thresholds allowed, we found differing results regarding the number of resources and thresholds required.

CHAPTER 4

PLAYING RENEWAL MULTI-RATE FLIPTHEM

In this chapter, we analyse the game of Multi-Rate (n,k)-FlipThem over the class of renewal strategies defined in Section 2.1. The player "renews" their play after every move, selecting the interval of time between each move independently and at random, over a fixed distribution. This interval of time is generated by a *renewal process*. No feedback about their opponent's moves is received and their move time only depends on their last move time. We begin by exploring the periodic strategy class, starting with the simple game of FlipIt and building it up to (n,k)-FlipThem. We then follow this same path for both the general renewal strategy class in Section 4.2 and the exponential strategy class in Section 4.3.

For the Periodic case, we reiterate a lot of the results found for FlipIt and FlipThem by the authors in [14, 27]. We supplement these results with benefit functions for Threshold FlipThem, generalising FlipIt and FlipThem. However, we reserve the proofs for the following sections when considering general renewal strategies. In Section 4.2, we fully derive the benefit functions for the completely general renewal threshold FlipThem. After this, we show the specific case of the periodic strategy class. Section 4.3 is based largely on our work presented in [17]. We show the specific benefit functions for the exponential strategy class and then derive Nash equilibria for all forms of the exponential game.

4.1 Periodic games

In this section, we look at the class of Periodic strategies. Whilst perhaps the simplest to define, to delve into this innocuous strategy class opens up a wealth of analytical intrigue. Interestingly, in [27] the authors show that the periodic strategy dominates the exponential strategy. So keep in mind that whilst this strategy seems overly simple, in the non-adaptive world it carries a lot of strength.

Here, we assume both players employ periodic strategies. By this, we mean a *periodic strategy with random phase*, characterised by a fixed interval δ between successive moves. We assume the first move time, named the phase, is chosen uniformly at random within the time interval $[0,\delta]$. The phase introduced is to remove the completely deterministic nature of the periodic strategy. Otherwise the opponent would have full information about the player and be able to schedule their move times to immediately after the player, thus controlling the resource completely. The average rate of play is denoted $\omega = 1/\delta$ and we define P_ω to be the periodic strategy with average rate of play ω and the class of all periodic strategies

$$\mathscr{P} = \{P_\omega | \omega > 0\}.$$

4.1.1 Playing periodic FlipIt

The first game we consider is FlipIt where the defender uses a periodic strategy P_ω and the attacker uses a periodic strategy P_σ. Denote $\delta^D = 1/\omega$ to be the period of play for the defender and $\delta^A = 1/\sigma$ to be the period for the attacker. We denote c_r^i to be the cost of moving on the resource $\mathscr{R}_r$ for Player $i \in \{A, D\}$. In Figure 4.1 we have a simulation of the FlipIt game with both players

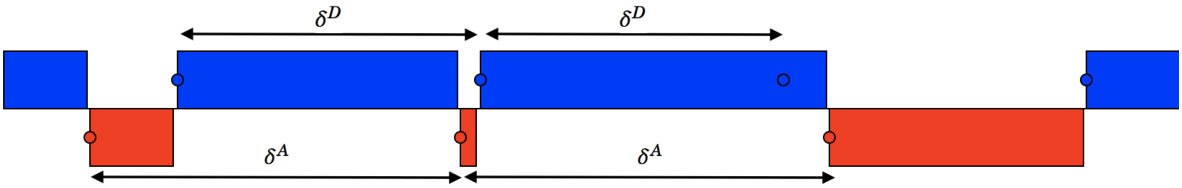


Figure 4.1: The FlipIt game played by a defender (blue) and attacker (red) both playing periodic strategies

employing periodic strategies. In this example the game is played over just 5 'seconds', with the defender (in blue) playing with a rate of $\omega = 1.1$ and the attacker a rate of $\sigma = 0.9$. This means that (ignoring the phase for now) every $\delta^D = 0.9$ 'seconds' the defender makes a move and every $\delta^A = 1.1$ 'seconds' the attacker makes a move. We see that the first moves for both players are indeed shorter as they are chosen uniformly at random in the intervals $[0,\delta^D]$ for the defender and $[0,\delta^A]$ for the attacker. Also, notice that the 3rd move for the defender is actually a wasted move since she is already in control of the resource at that point in time.

Benefit functions In the periodic FlipIt game described above, benefits for players depend on their rates of play ω and σ and their costs which we denote as c^D and c^A for the defender and attacker, respectively. In [27] the authors derive the benefit functions β^D and β^A for this particular game. Later, we prove a more general result for the whole class of renewal strategies. Therefore, we state their results without showing the derivation. For readers interested in the specific derivation for this result we direct them to [27].

Theorem 4.1. *In the non-adaptive game* FlipIt(P_ω, P_σ) *where both the defender and attacker play a periodic strategy with rate* ω *and* σ*, respectively. The benefit functions are:*
Case 1: $\omega \geq \sigma$ *(The defender plays faster or equal speed to the attacker)*

$$\beta^D(\omega,\sigma) = 1 - \frac{\sigma}{2\cdot\omega} - c^D\cdot\omega$$
$$\beta^A(\omega,\sigma) = \frac{\sigma}{2\cdot\omega} - c^A\cdot\sigma$$

Case 2: $\omega < \sigma$ *(The defender plays slower than the attacker)*

$$\beta^D(\omega,\sigma) = \frac{\omega}{2\cdot\sigma} - c^D\cdot\omega$$
$$\beta^A(\omega,\sigma) = 1 - \frac{\omega}{2\cdot\sigma} - c^A\cdot\sigma$$

Figure 4.2 displays heat plots showing the benefit functions of the rates of play ω and σ. We show these for different move costs c^D and c^A. The darker shaded areas correspond to a higher benefit received by the player and white areas correspond to zero or negative benefit. The periodic strategy is determined purely on the rates of play ω and σ. In the non-adaptive FlipIt game players are only aware of move costs for both players. Thus, players make a strategic decision based solely on the move costs of the game.

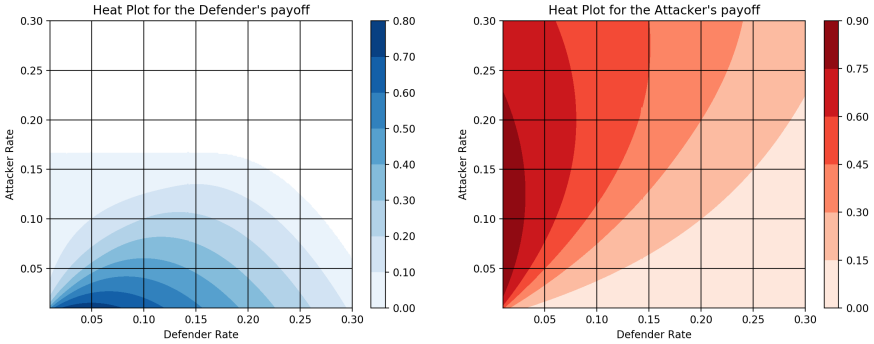
The top four graphs show that players have a significant advantage if their move costs are lower relative to their opponent, receiving higher benefit regardless of how their opponent plays. The bottom four graphs show the game for equal move costs between players. This shows that when move costs are equal, players receive similar benefits to their opponents and that increasing move costs negatively impacts players' benefits. We can also see that playing too fast can result in negative benefit (shown by the white areas of the graph).

Nash equilibria The periodic FlipIt game can be fully analysed to find the Nash equilibria, as was done in [27]. Once again, we state the authors' results and refer the reader to [27] for the specific proof.

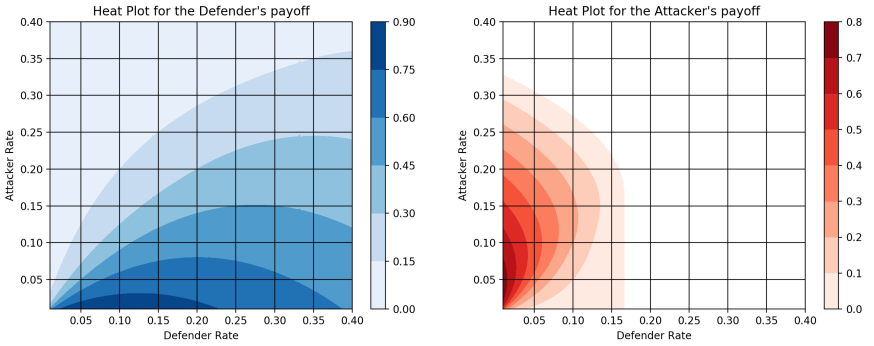
Theorem 4.2. *The non-adaptive, periodic game* FlipIt$(\mathscr{P},\mathscr{P})$*, in which the defender and attacker each play a strategy from the class of periodic strategies* $\mathscr{P}$*, has the Nash equilibria:*

$$c^D < c^A: \quad \alpha^{D*} = \frac{1}{2\cdot c^A}, \quad \alpha^{A*} = \frac{c^D}{2\cdot(c^A)^2},$$
$$c^D = c^A: \quad \alpha^{D*} = \alpha^{A*} = \frac{1}{2\cdot c^D},$$
$$c^D > c^A: \quad \alpha^{D*} = \frac{c^A}{2\cdot(c^D)^2}, \quad \alpha^{A*} = \frac{1}{2\cdot c^D}.$$

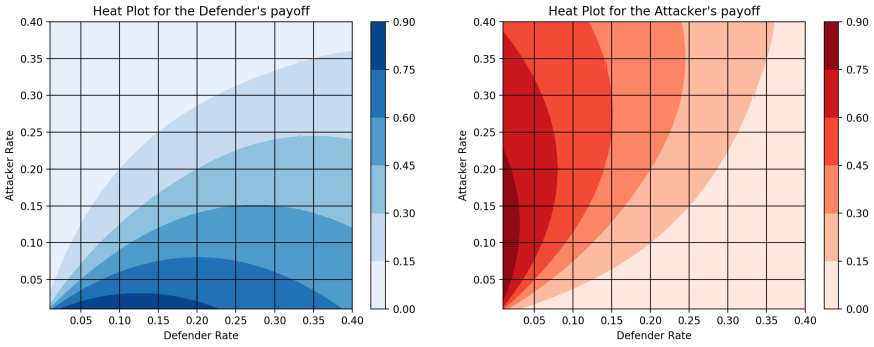
$c^D = 3$
$c^A = 1$



$c^D = 1$
$c^A = 3$



$c^D = 1$
$c^A = 1$



$c^D = 2$
$c^A = 2$

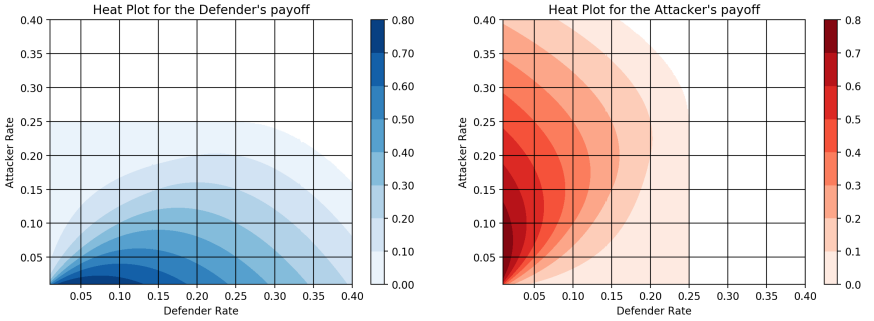


Figure 4.2: Heat plot of the non-adaptive periodic FlipIt game played by a defender (left) and attacker (right) with varying move costs.

4.1.2 Playing periodic FlipThem

In this section we extend the game to full threshold periodic FlipThem. This is played over the set of resources $\{\mathscr{R}_1,\dots,\mathscr{R}_n\}$ and players choose rates $P_{\omega_r},P_{\sigma_r} \in \mathscr{P}$ for the defender and attacker, respectively. Denote c_r^i to be the player move costs for each resource $\mathscr{R}_r$. Figure 4.3 shows an example of the defender (blue) and attacker (red) playing over 3 resources. The top three blue/red lines show the amount of time the players spend in control of each resource. Since this is a full threshold game, to receive gain (G^A) the attacker must be in control of all 3 resources simultaneously. In the bottom blue/red line, showing the overall state of the system, we see this between the black dashed lines. At any other time in which the attacker is not in control of all resources, the defender receives gain G^D. It should be easy to see that this scenario is now much harder for the attacker to receive a reasonable amount of benefit. He must move fast enough on every resource in order to be in control of all resources simultaneously meaning the number of moves required is higher. Therefore, if his move costs are high enough, the overall move costs of the system could quickly outweigh his gain, resulting in negative benefit.

Benefit functions In [14] the authors derive the benefit functions β^D and β^A for this particular game. Here, as in the FlipIt case, we state their results without showing the derivation as later on in this work we'll be showing more general results for the whole class of renewal strategies.

Theorem 4.3. *In the non-adaptive n-resource game* FlipThem$((P_{\omega_1},\dots,P_{\omega_n}),(P_{\sigma_1},\dots,P_{\sigma_n}))$ *where both the defender and attacker plays a periodic strategy with rate ω_r and σ_r on resource $\mathscr{R}_r$. The benefit functions are:*

$$\beta^D(S^D,S^A) = 1 - \left[\prod_{\substack{1\le r\le n,\\ \omega_r\ge\omega_r}} \frac{\sigma_r}{2\cdot\omega_r}\right]\left[\prod_{\substack{1\le r\le n,\\ \omega_r\le\omega_r}} 1-\frac{\omega_r}{2\cdot\sigma_r}\right] - \sum_{r=1}^{n} c_r^D\omega_r$$

$$\beta^A(S^D,S^A) = \left[\prod_{\substack{1\le r\le n,\\ \omega_r\ge\sigma_r}} \frac{\sigma_r}{2\cdot\omega_r}\right]\left[\prod_{\substack{1\le r\le n,\\ \omega_r\le\sigma_r}} 1-\frac{\omega_r}{2\cdot\sigma_r}\right] - \sum_{r=1}^{n} c_r^A\sigma_r$$

These benefits are long run average benefits. By this we mean we imagine the game is played for an infinite length of time multiple times and then averaged. If we were to check on the state of the system at a random point in time, since each resource is independent of one another the probability the attacker is in control of the whole system is the product of the probabilities he is in control of every resource. Thus, we start to get an intuition as to how to construct these benefit functions for various renewal strategies. This will be explored further and in a more technical manner in later sections.

Unfortunately, we have not yet managed to calculate explicitly a general Nash equilibrium for n resources. Of course, because we now have more than 2 dimensions to consider, we can no

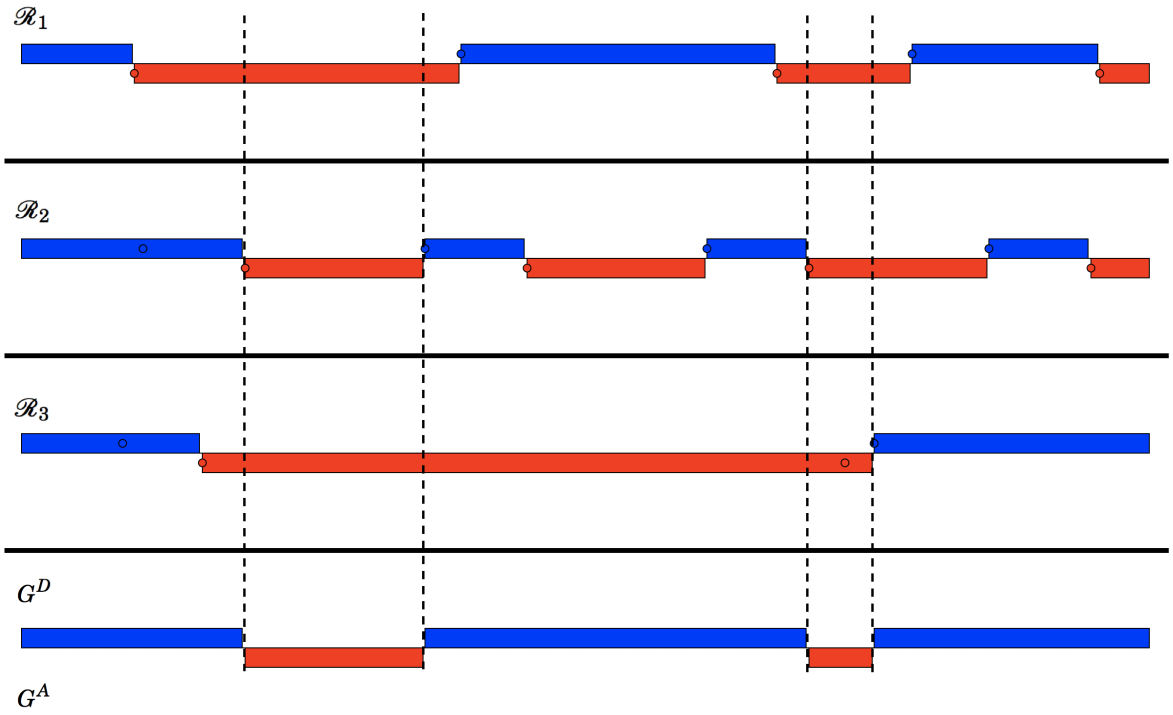


Figure 4.3: The Full Threshold FlipThem game played by a defender (blue) and attacker (red) both playing periodic strategies over 3 resources . Top three lines are the resources. The bottom shows the gain received by each player. To receive gain, the attacker must control all resources, shown between the dashed black lines.

longer show heat plots of this game. However, all is not lost as these benefit formulae will become useful in Chapter 6, when using genetic algorithms to find difficult-to-calculate equilibria.

4.1.3 Playing periodic (n,k)-FlipThem

Finally, we look at the game (n,k)-FlipThem for the periodic strategy class. Once again, this is played over the set of resources $\{\mathscr{R}_1,\ldots,\mathscr{R}_n\}$ and players choose rates $P_{\omega_r},P_{\sigma_r}\in\mathscr{P}$ for the defender and attacker, respectively. Denote c_r^i to be the player move costs for each resource $\mathscr{R}_r$. An example of this game is shown in Figure 4.4 played over 3 resources and with a threshold of 2. As above, the top 3 blue/red lines represent the amount of time the defender (blue) and attacker (red) spend in control of each resource. For the attacker to receive gain G^A, he must simultaneously be in control of 2 resources. We see this in the bottom blue/red line, between the first 2 black dashed lines. The attacker is in control of resource $\mathscr{R}_1$ and $\mathscr{R}_3$ (however not resource $\mathscr{R}_2$) and therefore receives gain in the bottom line.

In Figure 4.3 (full threshold case) and Figure 4.4 (partial threshold case), the rates of play and move costs are identical. It is easy to see the attacker is more successful in the partial case, being in control of the system for larger amounts of time. This starts to show what should be intuitively obvious, a lower threshold for the attacker produces a higher (lower for the defender) payoff for the same rate and move costs. Note that the attacker has to be in control of *at least* 2 resources. In other words, if he is in control of all 3 resources then he still receives some gain. Realise also that in this case, the defender must also be in control of at least 2 resources in order to achieve some gain. One can see this by inspecting Figure 4.4. So for this example, there is a nice symmetry to the game, as in the game of FlipIt.

Benefit functions Once again, we can derive benefit functions β^D and β^A for this particular game by applying a similar argument to the previous full threshold case and considering basic combinatorics. Here, as in both the previous cases, we state the results without showing the derivation as later on in this work we'll be showing more general results for the whole class of renewal strategies.

Theorem 4.4. *In the non-adaptive n resource game* (n,k)-FlipThem$((P_{\omega_1},\ldots,P_{\omega_n}),(P_{\sigma_1},\ldots,P_{\sigma_n}))$ *where both the defender and attacker play a periodic strategy with rate* ω_r *and* σ_r *on resource* $\mathscr{R}_r$.

$$\beta^D(S^D,S^A)=1-\sum_{\substack{C\subseteq\{1,\ldots,n\}\\|C|\geq k}}\left[\prod_{r\in C}\beta_r(\omega_r,\sigma_r)\right]\left[\prod_{r\notin C}\hat{\beta}_r(\omega_r,\sigma_r)\right]-\sum_{r=1}^{n}c_r^D\omega_r,$$

$$\beta^A(S^D,S^A)=\sum_{\substack{C\subseteq\{1,\ldots,n\}\\|C|\geq k}}\left[\prod_{r\in C}\beta_r(\omega_r,\sigma_r)\right]\left[\prod_{r\notin C}\hat{\beta}_r(\omega_r,\sigma_r)\right]-\sum_{r=1}^{n}c_r^A\sigma_r,$$

where

$$\beta_r(\omega_r,\sigma_r)=\begin{cases}1-\frac{\omega_r}{2\cdot\sigma_r} & \text{if } \omega_r<\sigma_r,\\ \frac{\sigma}{2\cdot\omega} & \text{if } \omega_r\geq\sigma_r,\end{cases}\qquad \hat{\beta}_r(\omega_r,\sigma_r)=\begin{cases}\frac{\omega}{2\cdot\sigma} & \text{if } \omega_r<\sigma_r,\\ 1-\frac{\sigma_r}{2\cdot\sigma_r} & \text{if } \omega_r<\sigma_r.\end{cases}$$

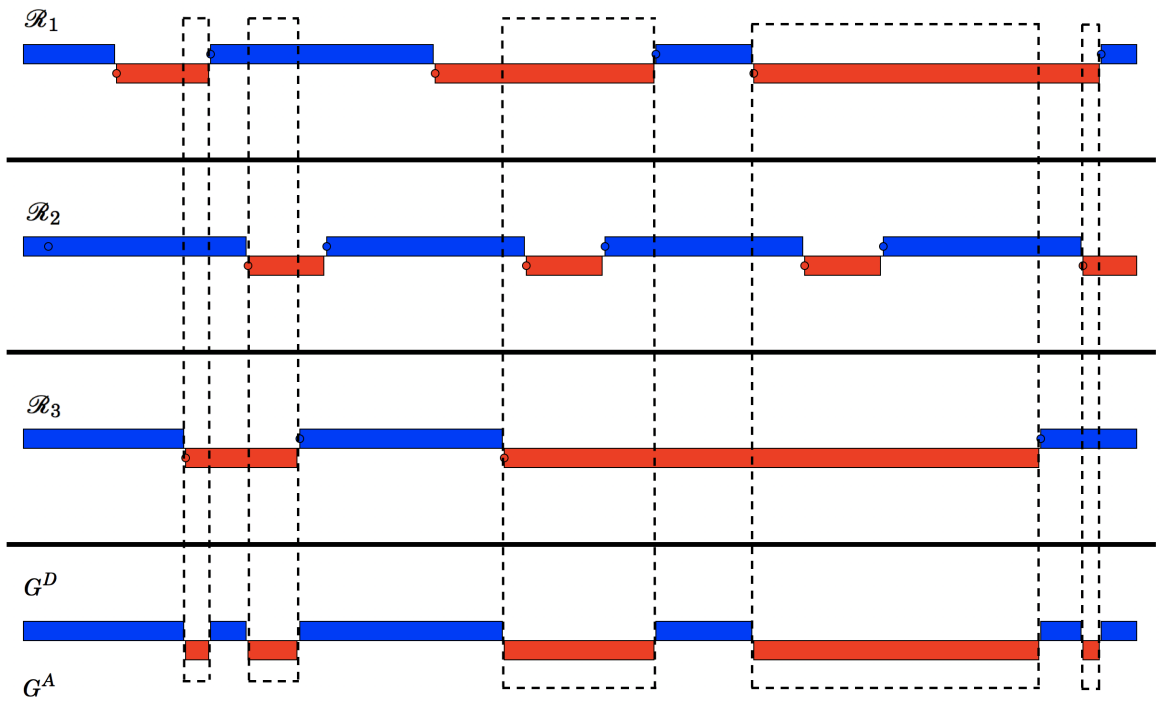


Figure 4.4: The Partial Threshold (3,2)-FlipThem game played by a defender (blue) and attacker (red) both playing periodic strategies over 3 resources. The top three lines are the resources. The bottom shows the gain received by each player. To receive gain, the attacker must control 2 out of 3 resources, shown between the dashed black lines.

4.2 General renewal strategies

In this section, we consider a general case of non-adaptive renewal strategies for both players. The inter-arrival times between each player's moves are generated by a renewal process based on a fixed probability distribution.

Following basic renewal theory [6], we let $\{X_j\}_{j\geq 0}$ be independent and identically distributed random variables chosen from a probability density function f with F being the corresponding cumulative density function. We can view $\{X_j\}_{j\geq 0}$ as the inter-arrival times between events in a renewal process with the n-th event arriving at time $S_n = \sum_{j=0}^{n} X_j$. Let $\mu = E(X_j)$ for all $j \geq 0$. The age function $Z(t)$ is defined to be the amount of time since the last event. i.e.

$$Z(t) = t - S_{N(t)}$$

where $N(t) = \sup\{n \geq 0 | S_n < t\}$. Denote $f_{Z(t)}$ and $F_{Z(t)}$ to be the density and cumulative distribution functions of the age function $Z(t)$. In [6] it is shown that

Lemma 4.1. *For a non-arithmetic renewal process, considering the age process $Z(t)$, the age*

density $f_Z(t)$ and cumulative distribution $F_Z(t)$ converge as:

$$f_Z(z) = \lim_{t\to\infty} f_{Z(t)}(z) = \frac{1-F(z)}{\mu},$$
$$F_Z(z) = \lim_{t\to\infty} F_{Z(t)}(z) = \frac{\int_0^z (1-F(x))\,\mathrm{d}x}{\mu}$$

where F is the cumulative distribution function of the inter-arrival times of the renewal process.

Playing FlipIt within the renewal strategy space The original FlipIt paper [27] defines the class of renewal strategies to be

$$\mathscr{R} = \{R_f | f \text{ is a non-arithmetic probability density function}\}$$

In other words, $\mathscr{R}$ is the class of all non-arithmetic renewal strategies. In this section we consider the FlipIt game such that both players are playing non-arithmetic renewal strategies. Thus, let us denote the inter-arrival times between each defender's move as $\{X_j\}_{j\geq 0}$ and the intervals between attacker's moves as $\{Y_j\}_{j\geq 0}$. Assume that $\{X_j\}_{j\geq 0}$ are identically distributed random variables generated independently from the probability density function f^D with expected value μ^D. Likewise, we assume $\{Y_j\}_{j\geq 0}$ are identically distributed random variables generated independently from the probability density function f^A with expected value μ^A. For each Player i, let $\alpha^i = 1/\mu^i$ be the rate of play. We denote the corresponding distribution functions as F^D and F^A for the defender and attacker, respectively. Since we have assumed both processes are non-arithmetic renewal process we can use Lemma 4.1 to find their age density and distribution functions. In [27] they use these to show that:

Theorem 4.5. *In the non-adaptive* FlipIt($\mathscr{R}\cup\mathscr{P}$) *game, the players' benefits for strategies* $(S^D, S^A) \in (\mathscr{R}\cup\mathscr{P})\times(\mathscr{R}\cup\mathscr{P})$ *are:*

$$\beta_D(S^D, S^A) = \int_{z=0}^{\infty} f_{Z^A}(z)\cdot F_{Z^D}(z)\,\mathrm{d}z - c^D\alpha^D,$$
$$\beta_A(S^D, S^A) = \int_{z=0}^{\infty} f_{Z^D}(z)\cdot F_{Z^A}(z)\,\mathrm{d}z - c^A\alpha^A$$

where f_{Z^i} and F_{Z^i} are the limits of the age density and cumulative distribution of the age process $Z^i(t)$ for Player $i \in \{A, D\}$.

Note that in Theorem 4.5, we have considered the FlipIt game within the strategy space $(\mathscr{R}\cup\mathscr{P})$. This includes the class of periodic strategies $\mathscr{P}$. This is a fantastic result for the renewal strategy space, and it is this that we build off in order to formulate benefit functions for our game (n,k)-FlipThem within the renewal strategy space. For a proof of theorem 4.5 please see [27].

Playing FlipThem within the renewal strategy space We now consider the game full threshold FlipThem within the renewal strategy space, including arithmetic strategies. In other

words, both renewal and periodic strategies. We consider the choice of strategy for Player i to be $S^i = (S^i_1, S^i_2, \ldots, S^i_n)$ where $S^i_r \in (\mathscr{R} \cup \mathscr{P})$ for all $1 \le r \le n$. In [14] they derive a general benefit function for this game. We explain their reasoning here and extend to prove the partial threshold case.

From theorem 4.5, the integral of each benefit function can be viewed as a proportion of time the player is in control of the system, or in that case, the one resource. Alternatively, we can view it as the probability the player is in control of the system at a random point in time. Therefore, within the FlipThem set up, the probability that the attacker is in control of resource $\mathscr{R}_r$ at any time $t > 0$ is

$$\mathbb{P}\left(Z^D_r(t) > Z^A_r(t)\right) = \int_{z_r=0}^{\infty} f_{Z^D_r}(z) \cdot F_{Z^A_r}(z_r)\,\mathrm{d}z_r$$

where $Z^i_r(t)$ is the length of time since Player i last moved on resource $\mathscr{R}_r$. All events occurring on various resources are independent of each other, meaning we can consider each resource to be independent. Thus,

$$\begin{aligned}&\mathbb{P}\left(Z^D_1(t) > Z^A_1(t),\ Z^D_2(t) > Z^A_2(t),\ \ldots,\ Z^D_n(t) > Z^A_n(t)\right)\\ &= \mathbb{P}\left(Z^D_1(t) > Z^A_1(t)\right) \cdot \mathbb{P}\left(Z^D_2(t) > Z^A_2(t)\right) \cdot \ldots \cdot \mathbb{P}\left(Z^D_n(t) > Z^A_n(t)\right)\end{aligned}$$

From this we have the result proved in [14]:

Theorem 4.6. *In the non-adaptive game* FlipThem$(\mathscr{R} \cup \mathscr{P})$*, with players' strategies* (S^D, S^A) *such that* $S^i = (S^i_1, S^i_2, \ldots, S^i_n)$ *and* $S^i_r \in (\mathscr{R} \cup \mathscr{P})$ *for* $i \in \{A, D\}$ *and* $1 \le r \le n$*, the players' benefit functions are:*

$$\begin{aligned}\beta_D(S^D, S^A) &= 1 - \prod_{r=1}^{n} \int_{z_r=0}^{\infty} f_{Z^D_r}(z) \cdot F_{Z^A_r}(z_r)\,\mathrm{d}z_r - \sum_{r=1}^{n} c^D_r \cdot \alpha^D_r,\\ \beta_A(S^D, S^A) &= \prod_{r=1}^{n} \int_{z_r=0}^{\infty} f_{Z^D_r}(z) \cdot F_{Z^A_r}(z_r)\,\mathrm{d}z_r - \sum_{r=1}^{n} c^A_r \cdot \alpha^A_r.\end{aligned}$$

Playing (n,k)-FlipThem within the Renewal Strategy Space Finally, we look at partial threshold (n,k)-FlipThem for n resources and threshold k for renewal strategies. Again, consider the choice of strategy for Player i to be $S^i = (S^i_1, S^i_2, \ldots, S^i_n)$ where $S^i_r \in (\mathscr{R} \cup \mathscr{P})$ for all $1 \le r \le n$. We state our theorem and follow with the proof.

Theorem 4.7. *In the non-adaptive game* (n,k)-FlipThem$(\mathscr{R} \cup \mathscr{P})$*, with players' strategies* (S^D, S^A) *such that* $S^i = (S^i_1, S^i_2, \ldots, S^i_n)$ *and* $S^i_r \in (\mathscr{R} \cup \mathscr{P})$ *for* $i \in \{A, D\}$ *and* $1 \le r \le n$*, the players' benefit functions are:*

$$\begin{aligned}\beta_D(S^D, S^A) &= 1 - \sum_{\substack{C \subseteq \{1,\ldots,n\}\\ |C| \ge k}} \left[\prod_{r \in C} \int_{z_r=0}^{\infty} f_{Z^D_r}(z) \cdot F_{Z^A_r}(z_r)\,\mathrm{d}z_r\right] \left[\prod_{r \notin C} \int_{z_r=0}^{\infty} f_{Z^A_r}(z) \cdot F_{Z^D_r}(z_r)\,\mathrm{d}z_r\right] - \sum_{r=1}^{n} c^D_r \alpha^D_r,\\ \beta_A(S^D, S^A) &= \sum_{\substack{C \subseteq \{1,\ldots,n\}\\ |C| \ge k}} \left[\prod_{r \in C} \int_{z_r=0}^{\infty} f_{Z^D_r}(z) \cdot F_{Z^A_r}(z_r)\,\mathrm{d}z_r\right] \left[\prod_{r \notin C} \int_{z_r=0}^{\infty} f_{Z^A_r}(z) \cdot F_{Z^D_r}(z_r)\,\mathrm{d}z_r\right] - \sum_{r=1}^{n} c^A_r \alpha^A_r.\end{aligned}$$

Proof: The probability that the attacker is in control of resource $\mathscr{R}_r$ at any time $t > 0$ is the probability the attacker moved more recently than the defender, or

$$\mathbb{P}\left(Z_r^D(t) > Z_r^A(t)\right)$$

where $Z_r^i(t)$ is the length of time since Player $i \in \{A, D\}$ last moved on resource $\mathscr{R}_r$. Likewise, the probability the defender is in control of resource $\mathscr{R}_r$ at any time $t > 0$ is

$$\mathbb{P}\left(Z_r^D(t) \leq Z_r^A(t)\right).$$

We need to know the probability the attacker receives gain. The threshold of the game is k, meaning the attacker receives gain whenever he is in control of at least $j \geq k$ resources. He is in control of j resources whenever he moved most recently on j out of n resources (and the defender most recently on the rest). We create a set made up of subsets of size j chosen from the set of resources $\{\mathscr{R}_1, \ldots, \mathscr{R}_n\}$. These are the possible states the game could be in such that the attacker is in control of j resources. Note that using the binomial coefficient, there are $\frac{n!}{j!(n-j)!}$ possible combinations of this. Therefore, the probability the attacker is in control of j resources at any time $t > 0$ is

$$\sum_{\substack{C \subseteq \{1,\ldots,n\} \\ |C|=j}} \left[\prod_{r \in C} \mathbb{P}\left(Z_r^D(t) > Z_r^A(t)\right)\right] \cdot \left[\prod_{r \notin C} \mathbb{P}\left(Z_r^D(t) \leq Z_r^A(t)\right)\right]$$

The threshold of the game is k, meaning the attacker receives gain whenever he is in control of at least $j \geq k$ resources. Therefore, the probability the attacker receives gain at any time $t > 0$ is

$$\sum_{\substack{C \subseteq \{1,\ldots,n\} \\ |C| \geq k}} \left[\prod_{r \in C} \mathbb{P}\left(Z_r^D(t) > Z_r^A(t)\right)\right] \cdot \left[\prod_{r \notin C} \mathbb{P}\left(Z_r^D(t) \leq Z_r^A(t)\right)\right].$$

From Theorem 4.5 we know the probability the attacker or defender is in control of resource $\mathscr{R}_r$ at any time $t > 0$ is

$$\mathbb{P}\left(Z_r^D(t) > Z_r^A(t)\right) = \int_{z_r=0}^{\infty} f_{Z_r^D}(z) \cdot F_{Z_r^A}(z_r)\,\mathrm{d}z_r, \text{ and}$$
$$\mathbb{P}\left(Z_r^D(t) \leq Z_r^A(t)\right) = \int_{z_r=0}^{\infty} f_{Z_r^A}(z) \cdot F_{Z_r^D}(z_r)\,\mathrm{d}z_r$$

The benefit of the attacker is simply the probability he is in control of the system minus his move costs. Thus

$$\beta_A(S^D, S^A) = \sum_{\substack{C \subseteq \{1,\ldots,n\} \\ |C| \geq k}} \left[\prod_{r \in C} \int_{z_r=0}^{\infty} f_{Z_r^D}(z) \cdot F_{Z_r^A}(z_r)\,\mathrm{d}z_r\right] \left[\prod_{r \notin C} \int_{z_r=0}^{\infty} f_{Z_r^A}(z) \cdot F_{Z_r^D}(z_r)\,\mathrm{d}z_r\right] - \sum_{r=1}^{n} c_r^A \alpha_r^A.$$

Lastly, the benefit of the defender is the probability the attacker isn't in control of the system minus her move costs.

$$\beta_D(S^D, S^A) = 1 - \sum_{\substack{C \subseteq \{1,\ldots,n\} \\ |C| \geq k}} \left[\prod_{r \in C} \int_{z_r=0}^{\infty} f_{Z_r^D}(z) \cdot F_{Z_r^A}(z_r)\,\mathrm{d}z_r\right] \left[\prod_{r \notin C} \int_{z_r=0}^{\infty} f_{Z_r^A}(z) \cdot F_{Z_r^D}(z_r)\,\mathrm{d}z_r\right] - \sum_{r=1}^{n} c_r^D \cdot \alpha_r^D \quad \square$$

We now have a completely general set of benefit functions for the renewal multi-rate threshold FlipThem game. Conveniently, we can go back and check our formulae for the periodic games by noting that the size bias density and cumulative distribution for the periodic strategy is

$$f_{Z^i}(z) = \begin{cases} \alpha^i & \text{if } z < \frac{1}{\alpha^i}, \\ 0 & \text{otherwise,} \end{cases} \qquad F_{Z^i}(z) = \begin{cases} \alpha^i \cdot z & \text{if } z < \frac{1}{\alpha^i}, \\ 1 & \text{otherwise.} \end{cases}$$

Therefore, if $\alpha_r^D < \alpha_r^A$ we have

$$\int_{z_r=0}^{\infty} f_{Z_r^D}(z) \cdot F_{Z_r^A}(z_r) \mathrm{d}z_r = \int_0^{\frac{1}{\alpha_r^A}} \alpha_r^D \cdot \alpha_r^A \cdot z \, \mathrm{d}z_r + \int_{\frac{1}{\alpha_r^A}}^{\frac{1}{\alpha_r^D}} \alpha_r^D \, \mathrm{d}z_r = 1 - \frac{\alpha^D}{2 \cdot \alpha_r^A}$$

Likewise, if $\alpha_r^D \geq \alpha_r^A$ then we have

$$\int_{z_r=0}^{\infty} f_{Z_r^D}(z) \cdot F_{Z_r^A}(z_r) \mathrm{d}z_r = \int_0^{\frac{1}{\alpha_r^D}} \alpha_r^D \cdot \alpha_r^A \cdot z \, \mathrm{d}z_r = \frac{\alpha^A}{2 \cdot \alpha_r^D}.$$

This is our definition of $\beta_r(\alpha^D, \alpha^A)$ in Theorem 4.4. Plugging the size bias density and distribution functions into $\int_{z_r=0}^{\infty} f_{Z_r^A}(z) \cdot F_{Z_r^D}(z_r) \mathrm{d}z_r$ for the two cases $\alpha_r^D < \alpha_r^A$ and $\alpha_r^D \geq \alpha_r^A$ will reveal $\hat{\beta}_r(\alpha^D, \alpha^A)$ in Theorem 4.4. The same can be done for Theorems 4.3 and 4.5.

4.3 Exponential games

In this section, we explore the (n,k)-FlipThem game whilst restricting our players to the exponential class strategy space $\mathscr{E}$. For ease of notation we assume that for each resource $\mathscr{R}_r$ the defender and attacker will play an exponential rate $\alpha_r^D = \mu_r$ and $\alpha_r^A = \lambda_r$, respectively. This means the move times for a player on a given resource $\mathscr{R}_R$ will follow a Poisson process with rate given by μ_r or λ_r.

Using Markov chain theory [9], we can construct explicit values for the proportion of time each player is in control of resource $\mathscr{R}_r$ depending on their rates of play. This stationary distribution is given by $\pi^r = (\pi_0^r, \pi_1^r) = \frac{1}{\mu_r + \lambda_r}(\mu_r, \lambda_r)$ and indicates that the defender is in control of the resource $\mathscr{R}_r$ a proportion $\mu_r/(\lambda_r + \mu_r)$ of the time, and the attacker is in control a proportion $\lambda_r/(\lambda_r + \mu_r)$ of the time. We assume that behaviour on each resource is independent of all the other resources and hence the proportion of time that the attacker is in control of a particular set of resources C, with the defender in control of the other resources, is simply given by the product of the individual resource proportions: $\prod_{r \in C} \frac{\lambda_r}{\lambda_r + \mu_r} \prod_{r \notin C} \frac{\mu_r}{\lambda_r + \mu_r}$. Conveniently, this matches up with our results from theorem 4.7 when using the result that the size-bias density and cumulative distribution functions for the exponential strategy are exactly the same as the exponential density and cumulative

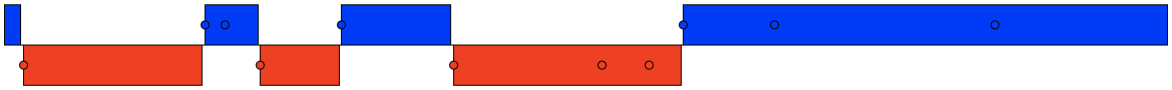


Figure 4.5: Simulation 1 of the FlipIt game played by a defender (blue) and attacker (red) both playing exponential strategies

Figure 4.6: Simulation 2 of the FlipIt game played by a defender (blue) and attacker (red) both playing exponential strategies

distribution. Thus, the benefit functions for the attacker and defender are given by

$$\begin{aligned}\beta_D(\boldsymbol{\mu},\boldsymbol{\lambda}) &= 1 - \sum_{\substack{C\subseteq\{1,\dots,n\}\\|C|\geq k}} \left[\prod_{r\in C}\frac{\lambda_r}{\lambda_r+\mu_r}\right]\cdot\left[\prod_{r\notin C}\frac{\mu_r}{\lambda_r+\mu_r}\right] - \sum_r c_r^D\cdot\mu_r \\ \beta_A(\boldsymbol{\mu},\boldsymbol{\lambda}) &= \sum_{\substack{C\subseteq\{1,\dots,n\}\\|C|\geq k}} \left[\prod_{r\in C}\frac{\lambda_r}{\lambda_r+\mu_r}\right]\cdot\left[\prod_{r\notin C}\frac{\mu_r}{\lambda_r+\mu_r}\right] - \sum_r c_r^A\cdot\lambda_r\end{aligned} \tag{4.1}$$

where the c_r^A and c_r^D are the (relative) move costs on resource $\mathscr{R}_r$ for attacker and defender, which are assumed fixed throughout the game, and $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ are the vectors of rates over all resources for the defender and attacker, respectively, constrained to be non-negative. The benefit functions in (4.1) show that the game is non-zero-sum, meaning we are unable to use standard zero-sum results found in the literature [21].

4.3.1 Playing exponential FlipIt

The first game we consider is FlipIt where both players use a strategy from the exponential strategy class $\mathscr{E}$. In figures 4.5 and 4.6 we have simulations of the FlipIt game with both players employing exponential strategies. Both simulations are played over just 5 'seconds' with exactly the same rates for defender (in blue) playing with a rate of $\mu = 1.0$ and the attacker a rate of $\lambda = 1.0$. This means that on average both players move every 'second'. However, that is as much information as we can determine about the next move. We can see from how different Figures 4.5 and 4.6 are that players' move times will vary significantly from game to game. However, if we extend this game to consider the long run proportions of time each player is in control of the system, these proportions are actually consistent and do not vary. This is different to the periodic game where the phase can be very important for an individual game.

4.3.2 Finding the equilibria of exponential (n,k)-FlipThem

In this section we analyse the game of multi-rate exponential (n,k)-FlipThem. We find a stationary point or equilibrium of the two benefit functions (4.1) in terms of purely the costs of moving on each resource. This would mean both players are playing at a rate that maximises their own benefit function with respect to their opponent's play and move costs. Thus, they would not wish to deviate from their current playing strategy or rates. This is known as a Nash Equilibrium [20], defined in Section 2.1.2.

Full threshold: multi-rate (n,n)-FlipThem We begin with the full threshold case in which the attacker must control all resources in order to obtain some benefit. The algebra is easier in this case; the partial threshold version is addressed in the following section. For this full threshold case, the general benefit functions (4.1) simplify to

$$\begin{aligned}\beta_D(\boldsymbol{\mu},\boldsymbol{\lambda}) &= 1-\prod_{r=1}^{n}\frac{\lambda_r}{\mu_r+\lambda_r}-\sum_{r=1}^{n}c_r^D\cdot\mu_r\\ \beta_A(\boldsymbol{\mu},\boldsymbol{\lambda}) &= \prod_{r=1}^{n}\frac{\lambda_r}{\mu_r+\lambda_r}-\sum_{r=1}^{n}c_r^A\cdot\lambda_r.\end{aligned} \tag{4.2}$$

We start by finding the best response function of the defender, which is a function br^D mapping attacker rates $\boldsymbol{\lambda}$ to the set of all defender rates $\boldsymbol{\mu}$ which maximise defender payoff β_D when the attacker plays rates $\boldsymbol{\lambda}$. A necessary, though not sufficient, condition for $\boldsymbol{\mu}$ to maximise β_D is that each μ_r maximises β_D conditional on the other values of $\boldsymbol{\mu}$. Furthermore, maxima with respect to μ_r occur either when the partial derivative $\frac{\partial\beta_D}{\partial\mu_r}$ is 0, or at a boundary of the parameter space. Equating this partial derivative to zero to gives

$$\frac{\partial\beta_D}{\partial\mu_r}=0\Rightarrow-\prod_{j=1}^{n}\lambda_j+c_r^D\cdot(\lambda_r+\mu_r)^2\cdot\prod_{j=1,j\neq r}^{n}(\mu_j+\lambda_j)=0.$$

This is a quadratic in μ_r, meaning that fixing the defender's benefit function shown in (4.2) for all attacker rates $\boldsymbol{\lambda}$ and all defender rates μ_j where $j\neq r$, gives only two turning points. Since β_D decreases to negative infinity as μ_r gets large, the two candidates for a maximising μ_r are at the upper root of this equation, or at $\mu_r=0$. A non-zero μ_r must therefore satisfy

$$\mu_r=-\lambda_r+\sqrt{\frac{\lambda_r}{c_r^D}\cdot\prod_{j=1,j\neq r}^{n}\frac{\lambda_j}{(\mu_j+\lambda_j)}}. \tag{4.3}$$

Of course, a μ_r satisfying this equation could be negative and thus inadmissible as a rate, but all we claim for now is that any non-zero μ_r must be of this form.

We can use the same method in differentiating the attacker's benefit with respect to his rate λ_r for resource $\mathscr{R}_r$, equating this to zero and manipulating to give

$$\lambda_r=-\mu_r+\sqrt{\frac{\mu_r}{c_r^A}\cdot\prod_{j=1,j\neq r}^{n}\frac{\lambda_j}{(\mu_j+\lambda_j)}}. \tag{4.4}$$

Any Nash equilibrium in the interior of the strategy space (i.e. with strictly positive rates on all resources) must be a simultaneous solution of (4.3) and (4.4). Note that both equations can be rearranged to express $\lambda_r + \mu_r$ in terms of a square root, and then we can equate the square root terms to find that $\frac{\lambda_r}{\mu_r} = \frac{c_r^D}{c_r^A}$. Substituting this relationship back in to (4.3) and (4.4) gives

$$\lambda_r^* = \frac{c_r^D}{(c_r^A + c_r^D)^2} \cdot \prod_{\substack{j=1 \\ j \neq r}}^{n} \frac{c_j^D}{(c_j^A + c_j^D)}, \quad \mu_r^* = \frac{c_r^A}{(c_r^A + c_r^D)^2} \cdot \prod_{\substack{j=1 \\ j \neq r}}^{n} \frac{c_j^D}{(c_j^A + c_j^D)}. \tag{4.5}$$

If a company was reviewing its defensive systems and was able to calculate these equilibria, they can see the rate at which they'd have to move in order to defend their system optimally, under the assumption that the attacker is also playing optimally. (Of course, if the attacker was playing sub-optimally, the defender would be able to gain a larger pay off.) From these equilibria the parties are able to calculate the long run proportion of time each of their resources within the system would be compromised by looking at the stationary distribution shown in Section 4.3. From this information, the company can also calculate the value of the benefit functions for each player when playing these rates. We do this by substituting the two equilibria back into the benefit functions (4.2). We can express these rewards in terms of the ratio between the costs of each resource, $\rho_j = \frac{c_j^A}{c_j^D}$, giving the dimensionless expression

$$\begin{aligned} \beta_D^* &= 1 - \prod_{r=1}^{n} \frac{1}{\rho_r + 1} \left[1 + \sum_{j=1}^{n} \frac{\rho_j}{\rho_j + 1} \right], \\ \beta_A^* &= \prod_{r=1}^{n} \frac{1}{\rho_r + 1} \left[1 - \sum_{j=1}^{n} \frac{\rho_j}{\rho_j + 1} \right]. \end{aligned} \tag{4.6}$$

We have expressed the payoffs to both players at the putative interior equilibrium. If this is an equilibrium, neither player will wish to deviate from these equilibrium rates. However, we have thus far ignored the potential maximising μ_r at 0. While no individual μ_r or λ_r will deviate unilaterally to zero (recall that the partial derivatives have only two zeroes, and thus payoff decreases as μ_r decreases to zero), if several rates simultaneously switch the payoff to a player could increase. We therefore consider what happens when rates can switch to zero, starting with the attacker.

By considering the attacker's benefit function (4.2), we can see that if the attacker plays a zero rate on any resource, then in order to maximise the payoff, she should play zero rates on the rest of the system. In other words, she should drop out of the game and receive zero reward. Thus by comparing the attacker payoff in (4.6) to zero we can see quickly whether this is indeed a point at which the attacker will be content.

For the defender, things are more complicated. However, we can see from the benefit function (4.2) that a zero payoff by withdrawing from the game is again an option, and so we compare the equilibrium payoff (4.6) to zero as a partial check on whether the fixed point (4.5) is an equilibrium. We do not explicitly discount partial dropout of the defender, but note that by dropping out the

defender is effectively reducing the game to a smaller number of servers, and hence should not have invested in the additional servers in the first place. Comparing the benefits in (4.6) to zero it is easy to see in this full threshold case that the defender's benefit β_D^* is always non-negative when playing (4.5) meaning dropping out is not an equilibrium point for the defender, whereas β_A^* can drop below 0. We use this to find a condition which must be satisfied for the point (4.5) to be an equilibrium. In particular, we require β_A^* in (4.6) to be positive, and therefore

$$1 - \sum_{j=1}^{n} \frac{\rho_j}{\rho_j + 1} > 0. \tag{4.7}$$

Thus, we have a condition (4.7) that, if satisfied, the attacker will not drop out of the game. If the condition is not satisfied, the attacker will prefer to drop out of the game entirely than to play at the interior equilibrium. Ensuring condition (4.7) is met can thus be viewed as a design criterion for system engineers when designing defensive systems.

Partial threshold: multi-rate (n,k)-FlipThem So far we have extended the full threshold FlipThem game [14] by obtaining the equilibria of the benefit functions constructed from the proportion of time the attacker is in control of the whole system. In order to generalise this further, we return to our partial threshold case in which the attacker gains benefit from controlling only $k < n$ resources. Our general benefit functions for both players are written in (4.1); the analysis is analogous to methods demonstrated previously. In this (n,k)-threshold case, the analogous best response conditions to (4.3) and (4.4) are

$$\mu_r = -\lambda_r + \sqrt{\frac{\lambda_r \cdot S_r}{c_r^D}} \quad \text{and} \quad \lambda_r = -\mu_r + \sqrt{\frac{\mu_r S_r}{c_r^A}}, \tag{4.8}$$

where we have introduced S_r to denote

$$\sum_{\substack{C \subseteq \mathscr{A}_i \\ |C| \geq k}} \left[\prod_{j \in C} \frac{\lambda_j}{\lambda_j + \mu_j} \right] \cdot \left[\prod_{j \notin C} \frac{\mu_j}{\lambda_j + \mu_j} \right]$$

and $\mathscr{A}_r = \{1, \ldots, r-1, r+1, \ldots, n\}$. Interestingly, equating the square root terms results in the same relationship $\frac{\mu_r}{c_r^A} = \frac{\lambda_r}{c_r^D}$ as in Section 4.3.2. Finally, substituting this relationship back into the best response functions (4.8) gives us

$$\begin{aligned} \mu_r^* &= \frac{c_r^A}{(c_r^A + c_r^D)^2} \cdot \sum_{\substack{C \subseteq \mathscr{A}_i \\ |C| \geq k}} \left[\prod_{j \in C} \frac{c_j^D}{c_j^A + c_j^D} \right] \cdot \left[\prod_{j \notin C} \frac{c_j^A}{c_j^A + c_j^D} \right], \\ \lambda_r^* &= \frac{c_r^D}{(c_r^A + c_r^D)^2} \cdot \sum_{\substack{C \subseteq \mathscr{A}_i \\ |C| \geq k}} \left[\prod_{j \in C} \frac{c_j^D}{c_j^A + c_j^D} \right] \cdot \left[\prod_{j \notin C} \frac{c_j^A}{c_j^A + c_j^D} \right]. \end{aligned} \tag{4.9}$$

As in Section 4.3.2, (4.9) is a Nash equilibrium for the game, unless one or other player can improve their payoff by dropping out of one or more resources. We can substitute these rates back

into the players' benefit functions as we did in the full threshold case in Section 4.3.2 to check that the payoffs at this putative equilibrium are non-negative. However, the formulae become very complicated to write down explicitly in general and we leave this to Chapter 5, when we deal with specific examples. Note this also means we do not have a clean condition analogous to (4.7) to test whether (4.9) is an equilibrium.

4.4 Summary

In this chapter we analysed the game of multi-rate (n,k)-FlipThem over the class of renewal strategies defined in Section 2.1. In Section 4.2, we derived the benefit functions for the general renewal threshold FlipThem game, generalising results from [14, 27] to the partial threshold FlipThem game. In Section 4.3, we showed the specific benefit functions for the exponential strategy class and then derived Nash equilibria for all forms of the exponential game, providing analysis of optimum strategies for the defender and attacker based of the ratio between their move costs on each resource. We derived the inequality (4.7) that when met will ensure the attacker does not drop out of the game. Thus, conditions such as these can be viewed as a design criterion when designing defensive systems.

CHAPTER 5

FICTITIOUS PLAY IN MULTI-RATE (n,k)-FLIPTHEM

While the benefit and equilibrium analysis above offers useful insight into the security game multi-rate threshold FlipThem, it can be viewed as an unrealistic model of real world play. In particular, it is extremely unlikely the players have full knowledge of the benefit and move costs of their opponent, and therefore cannot calculate the equilibrium strategies. In section 5.1 we remove this assumption, only allowing players to know their own benefit functions and at the end of a defined point in time players receive the number of moves made by their opponent. We also assume that players use a strategy from the exponential strategy class. The players apply what is commonly known in game theory literature as *fictitious play*. This chapter is based largely on our work presented in the fictitious play section of [17].

5.1 Introducing fictitious play into multi-rate (n,k)-FlipThem

We now introduce game-theoretical learning, in which the only knowledge a player has of their opponent is the actions that they take. When the game is repeatedly played through time, players respond to their observations and attempt to improve their payoff. In this section, we focus on a method of learning known as *fictitious play* [2, 4, 8].

We break the game up into periods of fixed length of time. At the end of period τ each player observes the number of times the button of each resource r was pressed by their opponent in that period. Denote by λ_r^τ and μ_r^τ the actual rate played by attacker and defender in period τ, and use $\widetilde{\lambda_r}^\tau$ and $\widetilde{\mu_r}^\tau$ to denote the number of button presses by the attacker and defender, respectively. After $\mathcal{T}$ plays of the game, each player averages the observations they have made of their opponent, resulting in estimates for each resource

$$\widehat{\lambda}_r^{\mathcal{T}} = \frac{1}{\mathcal{T}} \sum_{\tau=1}^{\mathcal{T}} \widetilde{\lambda}_r^\tau, \qquad \widehat{\mu}_r^{\mathcal{T}} = \frac{1}{\mathcal{T}} \sum_{\tau=1}^{\mathcal{T}} \widetilde{\mu}_r^\tau.$$

The players select their rates for time period $\mathscr{T}+1$ by playing a best response to their estimates;

$$\mu_r^{\mathscr{T}+1} = \mathrm{br}_r^D(\widehat{\boldsymbol{\lambda}}^{\mathscr{T}}), \qquad \lambda_r^{\mathscr{T}+1} = \mathrm{br}_r^A(\widehat{\boldsymbol{\mu}}^{\mathscr{T}}).$$

where $\widehat{\boldsymbol{\mu}}^{\mathscr{T}}$ and $\widehat{\boldsymbol{\lambda}}^{\mathscr{T}}$ are the defender and attacker's vector of rates played on each resource. If it were the case that opponent rates were constant, the averaging of observations over time would be an optimal estimation of those rates. Since both players are learning, the rates are not constant, and averaging uniformly over time does not result in statistically optimal prediction. However lack of a better informed model of rate evolution means that averaging is retained as the standard mechanism in fictitious play; see [18, 26] for attempts to move beyond this assumption.

```
Set maximum periods to be played;
Randomly assign move costs for each player;
Randomly choose initial starting rates for both players;
One period is played;
while Number of periods has not reached the maximum do
    Each player receives last period's observation of their opponent's play;
    Each player averages their opponent's history to form a belief;
    Each player uses their best response function, given this belief, to calculate their rate for
      the next period;
    Each player plays this rate;
end
```

Algorithm 1: Fictitious Play algorithm for multi-rate (n,k)-FlipThem

This fictitious play process is described in Algorithm 1, where we see the simplicity of the learning process and the sparsity of the information required by the players. The only challenging step is in calculating the best response function. As observed in Section 4.3.2, the best response of each player is not, in general, a simple analytical function of the rates of the opponent.

From the defender's point of view we consider all subsets of the resources, setting the rates of these resources to zero and solving (4.8) for the non-zero rates. We define the best response as the set of rates with the highest payoff given the fixed belief $\widehat{\boldsymbol{\lambda}}^{\mathscr{T}}$. The attacker's best response is calculated analogously. An interesting question, which we address, is whether this simple learning process converges to the equilibria calculated previously in Section 4.3.2.

The process we have defined is a discrete time stochastic process. It is in actual fact a stochastic fictitious play process [7]; since the number of button presses in a period is Poisson with expected value the played rate multiplied by the length of the time interval, the observations can be seen to satisfy

$$\widetilde{\mu}_r^{\mathscr{T}+1} = \mathrm{br}_r^D(\widehat{\boldsymbol{\lambda}}^{\mathscr{T}}) + M_{\mu,r}^{\mathscr{T}+1}, \qquad \widetilde{\lambda}_r^{\mathscr{T}+1} = \mathrm{br}_r^A(\widehat{\boldsymbol{\mu}}^{\mathscr{T}}) + M_{\lambda,r}^{\mathscr{T}+1},$$

where $\mathbb{E}(M_{\cdot,\cdot}^{\tau+1}|\mathscr{F}^{\tau}) = 0$ if $\mathscr{F}^{\tau}$ denotes the history of the process up to time τ. The methods of [2] then apply directly to show that the convergence (or otherwise) of the discrete stochastic process

is governed by the continuous deterministic differential equations

$$\frac{\mathrm{d}\boldsymbol{\lambda}}{\mathrm{d}t} = \mathrm{br}(\boldsymbol{\mu}) - \boldsymbol{\lambda}, \qquad \frac{\mathrm{d}\boldsymbol{\mu}}{\mathrm{d}t} = \mathrm{br}(\boldsymbol{\lambda}) - \boldsymbol{\mu}. \tag{5.1}$$

In standard fictitious play analyses, one might show that solutions of (5.1) are globally convergent to the equilibrium set. This is commonly achievable only in some classes of games, and since we do not have a zero-sum game we have not been able to show the required global convergence of (5.1). However, we find interesting results by simulating Algorithm 1. We explore this in the rest of the chapter.

5.1.1 Original FlipIt

The game of FlipIt can be considered a special case of our game of multi-rate (n,k)-FlipThem, seen by setting $n = k = 1$. This has the advantage that the best responses can be written in closed form, and we can use (5.1) to set up a two dimensional ordinary differential equation in terms of the players' rates and time. We start by writing the benefit functions for this special case

$$\beta_D(\mu,\lambda) = 1 - \frac{\lambda}{\mu+\lambda} - d\mu, \qquad \beta_A(\mu,\lambda) = \frac{\lambda}{\mu+\lambda} - a\lambda. \tag{5.2}$$

Differentiating the players' benefit functions (5.2) in terms of their respective resource rates and then solving for these gives the best response functions

$$\mathrm{br}^D(\lambda) = \left(-\lambda + \sqrt{\frac{\lambda}{d}}\right)^+, \qquad \mathrm{br}^A(\mu) = \left(-\mu + \sqrt{\frac{\mu}{a}}\right)^+$$

where $(x)^+ = \max(x,0)$. The ordinary differential equation (5.1) becomes

$$\frac{\mathrm{d}\lambda}{\mathrm{d}t} = \left(-\mu + \sqrt{\frac{\mu}{a}}\right)^+ - \lambda, \qquad \frac{\mathrm{d}\mu}{\mathrm{d}t} = \left(-\lambda + \sqrt{\frac{\lambda}{d}}\right)^+ - \mu. \tag{5.3}$$

This is a two dimensional ordinary differential equation in terms of the players' rates and time. The plot of the phase portrait of this is shown in Figure 5.1. where we have used $d = 0.1, a = 0.3$ as the move costs. It easy to see that the arrows demonstrating the direction of the rates over time converge upon a single point. This point is the equilibrium we can calculate easily from the more general equilibria derived in Section 4.3.2. We can also use Algorithm 1 to plot trajectories of the system in order to view convergence of the system; the convergence is monotonic and uninteresting so we omit the plots.

5.1.2 Single rate (n,k)-FlipThem

A multi-resource example in which we retain one rate per player (as opposed to one rate per resource for each player) is given by the situation in Chapter 3; each player chooses a rate at which to play all resources. Whilst this allows us to retain a two-dimensional system when

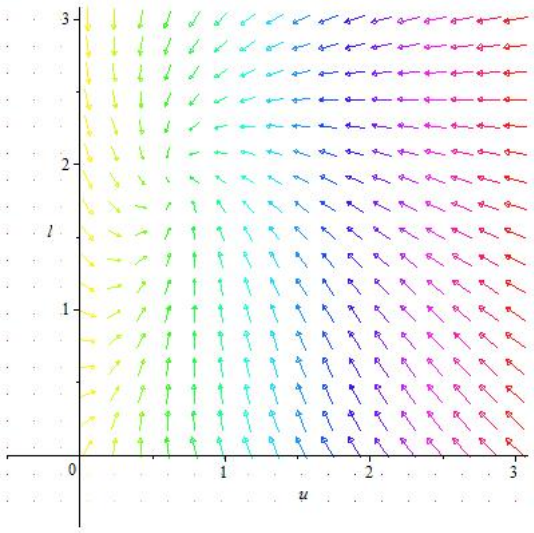


Figure 5.1: Phase Portrait of (5.3) with $d = 0.1, a = 0.3$

considering the multiple resource case, unfortunately obtaining explicit best response functions is extremely difficult. Therefore, we revert to Algorithm 1 using time intervals of length 100; we fix this time interval for all further experiments in this Chapter.

As in those previous works, we consider the stationary distribution of the system, with the defender playing with rate μ and attacker rate λ. This results in a stationary distribution for the whole system, given by

$$\pi = \frac{1}{(\mu+\lambda)^n}\left(\mu^n, n\cdot\lambda\cdot\mu^{n-1}, \ldots, \binom{n}{k}\cdot\mu^{n-k}\cdot\lambda^k, \ldots, n\cdot\mu\cdot\lambda^{n-1}, \lambda^n\right),$$

where the states correspond to the number of compromised resources, ranging from 0 to n. Benefit functions for both players are given by

$$\beta_D(\mu,\lambda) = 1 - \sum_{r=k}^{n}\pi_r - d\cdot\mu = 1 - \frac{1}{(\mu+\lambda)^n}\cdot\sum_{r=k}^{n}\binom{n}{r}\cdot\mu^{n-r}\cdot\lambda^r - d\cdot\mu,$$
$$\beta_A(\mu,\lambda) = \sum_{r=k}^{n}\pi_r - a\cdot\lambda = \frac{1}{(\mu+\lambda)^n}\cdot\sum_{r=k}^{n}\binom{n}{r}\cdot\mu^{n-r}\cdot\lambda^r - a\cdot\lambda,$$

and best responses are calculated by differentiating these benefit functions, as in [16] and Section 4.3.2. In Figure 5.2, we plot the rate of the attacker and defender respectively by applying Algorithm 1 to random starting rates for both players, with $(3,2)$-threshold and costs $a = 0.65$ and $d = 0.5$. The straight horizontal lines represent the players' Nash equilibrium rates that can be calculated as a special case of the general equilibrium from (4.9). Note that we have chosen these costs in order to produce positive benefits for both players whilst playing the calculated Nash equilibrium. We see that both the defender's and attacker's mean rate converges to these lines.

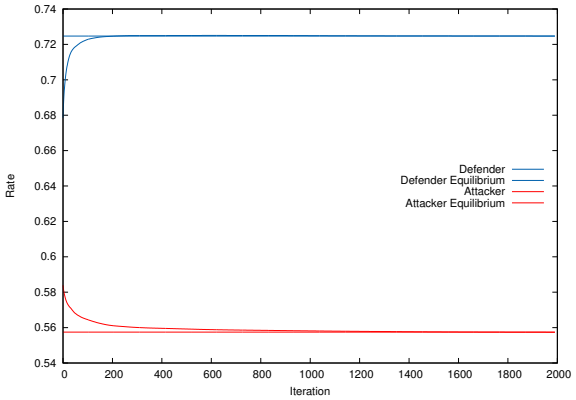


Figure 5.2: Mean of the defender's and attacker's rate with $(3,2)$-threshold and ratio $\rho = \frac{a}{d} = 1.3$.

5.1.3 Multi-rate (n,k)-FlipThem

Finally, we come to the most general case of the chapter, multi-rate (n,k)-FlipThem. As observed in Chapter 4, depending on the player's belief of their opponent's rates on each resource, they may choose to drop out of playing on a certain resource, or perhaps even all of them. Our best response functions must iterate through all possibilities of setting the resource rates to zero and chooses the configuration with the highest benefit as the best response. This solution is then used as $\mu^{\mathcal{T}+1}$ or $\lambda^{\mathcal{T}+1}$ for the following period $\mathcal{T}+1$.

We want to find a condition such that we can gain some insight as to whether in this most complicated setting our iterative learning rule will converge to the equilibria we calculated analytically in Section 4.3.2. We experimented with multiple combinations of n and k, randomly simulating costs of both players. From these empirical results, we observe that convergence occurs whenever the internal fixed point (4.9) results in non-negative benefits to both players.

Specific case $(n=3, t=2)$**:** In order to illustrate the outcomes of our fictitious play algorithm we specify our threshold $(n,k) = (3,2)$, and choose two representative cases of our randomly simulated examples. These particular examples were selected for ease of display, allowing us to illustrate their properties of convergence (or divergence) clearly on just one figure. The first, when the benefits are positive and the internal equilibrium is not ruled out, we term 'success'. The second is an example in which the internal fixed point is not an equilibrium, and we term this case 'failure'.

Success: Our first example is with ratios $(\rho_1,\rho_2,\rho_3) \approx (0.7833, 0.7856, 0.7023)$ and attacker costs $(a_1,a_2,a_3) \approx (0.6237, 0.5959, 0.5149)$. Thus, the benefit functions from equation (4.1) in section 4.3 at equilibrium are $\beta_D^* \approx 0.0359$ and $\beta_A^* \approx 0.2429$. Since we have positive payoff for both players we expect convergence of the learning algorithm. This is exactly what we observe

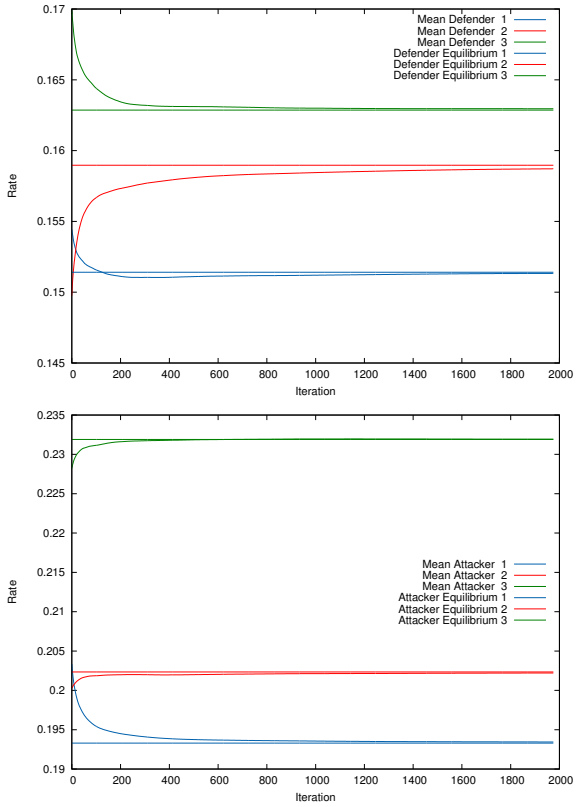


Figure 5.3: Mean of the defender's rates (top) and attacker's rates (bottom) on all resources with $(3,2)$-threshold and ratios $(\rho_1,\rho_2,\rho_3) \approx (0.7833, 0.7856, 0.7023)$

in Figure 5.3; convergence of rates to the lines representing the equilibria calculated in Section 4.3.2.

Failure: Our second example shows a lack of convergence when conditions are not met. Therefore, we choose ratios to be $(\rho_1,\rho_2,\rho_3) \approx (0.5152, 0.5074, 0.5010)$ and attacker costs $(a_1,a_2,a_3) \approx (0.2597, 0.2570, 0.2555)$. This gives 'equilibrium' benefits for the defender and attacker of $\beta_D^* \approx -0.0354$ and $\beta_A^* \approx 0.4368$. The defender's benefit in this situation is negative, meaning we expect a lack of convergence. Figure 5.4 shows the development of both players' rates over time. We can see the rates do not approach anywhere near the equilibrium. Intuitively, this makes sense as the defender will not choose to end up playing in a game in which she receives negative payoff when she can drop out of the game completely in order to receive zero payoff.

We can see evidence of this dropping out in Figure 5.5, which shows the rates the defender is actually playing on the resource (rather than the mean over time). The defender has certain periods where she drops out of the game entirely. The attacker's mean rates then start to drop until a point when the defender decides to re-enter the game.

To see a reason for this volatility, Figure 5.6 shows the defender's payoff surface for nearby attacker strategies on either side of a 'dropout' event from Fig. 5.5. We fix the defender's rate on

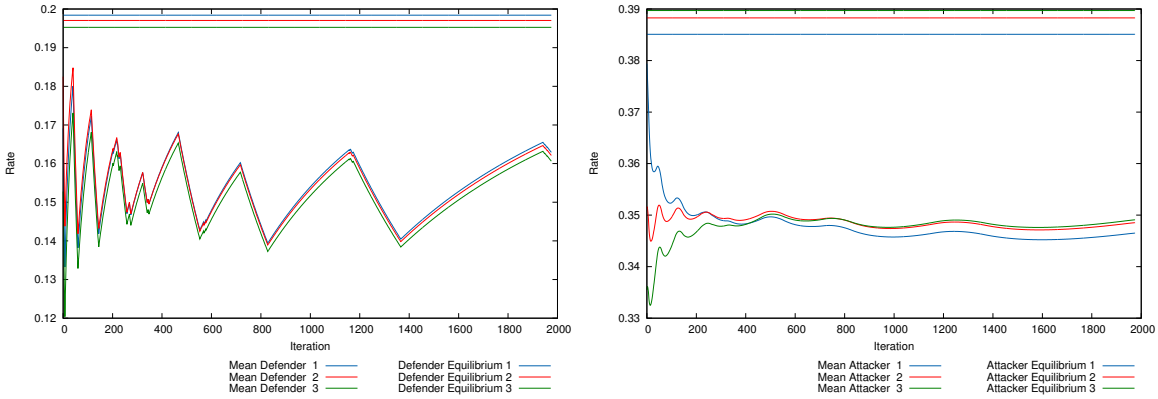


Figure 5.4: Mean of the defender's rates (top) and attacker's rates (bottom) on all resources with $(3,2)$-threshold and ratios $(\rho_1,\rho_2,\rho_3) \approx (0.5152, 0.5074, 0.5010)$.

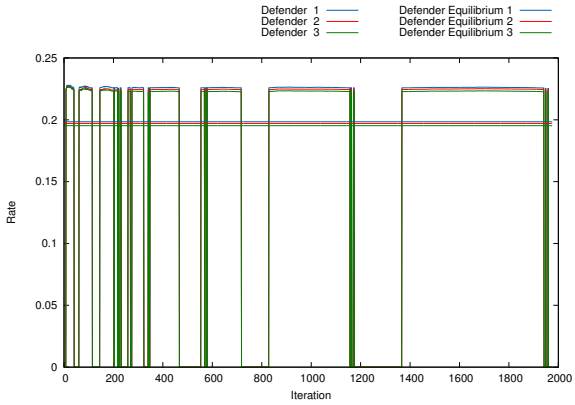


Figure 5.5: Defender's actual rates on all resources with $(3,2)$-threshold and ratios $(\rho_1,\rho_2,\rho_3) \approx (0.5152, 0.5074, 0.5010)$.

resource 1 and plot the benefit surface as the rates on resources 2 and 3 vary. We also plot the plane of zero reward. It's easy to observe that the maximum of this reward surface is above 0 in the left hand plot, but a small perturbation of the attacker rates pushes the maximal defender benefit below zero in the right hand plot, thus forcing the defender to drop out entirely. We conjecture that as the ratios decrease (and therefore become more costly for the defender) the defender drops out of the game more often within this learning environment.

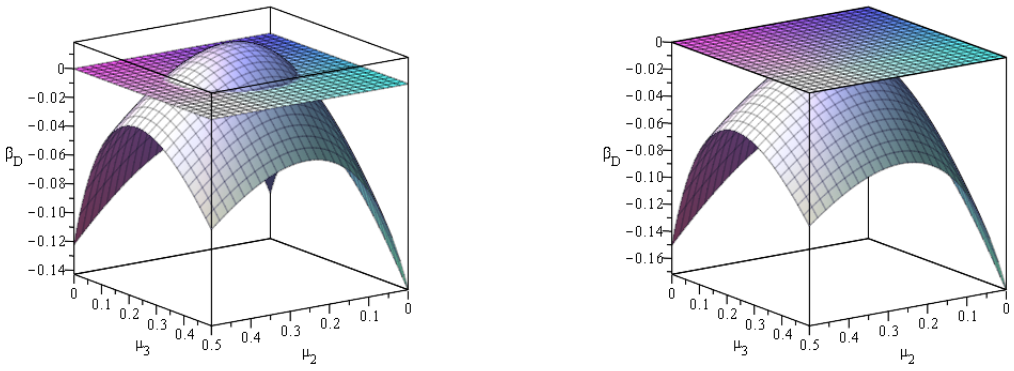


Figure 5.6: Two snapshots of the defender's reward surface with a slight perturbation in attackers rates. (Left) The Payoff is just above the 0-plane. (Right) The whole payoff surface is below the 0-plane.

5.1.4 Summary

In this chapter, we introduced fictitious play as a form of learning within the Threshold FlipThem game. This has removed the assumption that players are aware of their opponent's payoff functions and move costs within the game. The game was broken up into periods of fixed lengths of time, at the end of which, players observe their opponent's number of moves and respond accordingly. We have found that under conditions calculated in Chapter 4, player's rates converge to equilibria also calculated in Chapter 4.

CHAPTER 6

GENETIC ALGORITHMS

In the previous chapter, we introduced a method of learning named fictitious play. This made the scenario more realistic by removing the assumption that players are aware of their opponent's payoff functions and move costs within the game. In other words, players are only aware of their own payoff functions and move costs.

In this chapter, we go one step further by completely removing the assumption that players know analytically their payoff functions and move costs. To do this, we use genetic algorithms (GAs). Developed by Holland [12] in 1975, they have been used as a tool to find high-quality, optimal solutions to problems spanning multiple dimensions and hence producing a large solution space too unwieldy to be discovered by conventional search methods. This GA method allows us to create a population of strategies for each player and evolve the population to update over many generations to see if the population converges to the optimum strategy calculated in previous papers. In each generation, strategies from the defender's population are paired up against strategies from the attacker's population in order to play the game. The only value that is returned is the specific payoff for that game. A fraction of the available games are played and the average payoff for each population strategy is taken. These are then ranked within the population and a special process is used to decide what makes up the following population generation (to be discussed below).

The advantage of this approach is we have removed even more assumptions of player knowledge. This is useful in the real world when defence specialists are actually unaware of who they're playing, what their motivations are, or how the adversary is deciding to infiltrate the system. With just a small amount of information, the specialists are able to make decisions that optimise their security.

Here, we consider two experiments. The first experiment involves just one strategy class

within the game, we find that in certain strategy classes the average of the top performing strategies within a population converge to the Nash equilibrium calculated analytically in Chapter 4. This helps validate our approach via genetic algorithms as it implies we subsume the prior analytic approach in our models. The second experiment involves mixed strategies within a single population. We find that if a strictly dominant strategy class from within the population exists, then this strategy will overpower all other strategies. Thus, we are able to compare strategies even when analytically this may be impossible.

Contributions

- We develop a GA that converges to Nash equilibrium for certain analysed games
- We apply the GA to scenarios with multiple strategy classes to discover dominant strategies
- We have produced a python library to test new strategies within the multi-rate FlipThem space

6.1 Model

We now describe how we formulate our genetic algorithm to explore strategy spaces within the game of multi-rate threshold FlipThem. We denote the defender to be D and attacker A.

Initialising population Choose two populations of equal size N consisting of genes. One population is deemed to be the defender population and the other the attacker population. A gene g is defined to be a pair $(\boldsymbol{s}, \boldsymbol{\alpha})$ such that $\boldsymbol{s} = (s_1, \ldots, s_N)$ is a vector of strategies across resources and, for all j, we have $s_j \in \mathscr{R} \cup \mathscr{P}$ is a strategy class in the strategy space consisting of Periodic strategies $\mathscr{P}$ and non-arithmetical renewal strategies $\mathscr{R}$. Likewise, $\boldsymbol{\alpha} = (\alpha_1, \ldots \alpha_N)$ is a vector of rates such that $\alpha_j \in [0, \infty)$ is the rate of play associated with the given strategy s_j for all j.

Game A game is played between two genes, one chosen from each population. The multi-rate FlipThem game is played between the two genes giving the benefits $B_D(g_D, g_A)$ and $B_A(g_D, g_A)$ for the defender gene g_D and attacker gene g_A. These are simulated with our library. Each game consists of a specific number of rounds k to be played between the two chosen genes of the population. The average result of the rounds played is then taken, such that

$$B_D = \frac{1}{k} \sum_{i=1}^{k} B_D^i(g_D, g_A),$$
$$B_A = \frac{1}{k} \sum_{i=1}^{k} B_A^i(g_D, g_A).$$

This ensures an accurate result is recorded when two stochastic strategies are simulated. Playing just one round could give an unrepresentative result.

Tournament play If both populations chosen have size N, then there are N^2 available *games* available to be played between populations. Any knowledge of quadratic functions will show this increases with N so we choose a proportion of these games to be played in order to speed up computation. The specific games to be played are chosen at random, without replacement (in other words, once a game is played, it cannot be played again). The tournament is a simulation of each game. This means we can create any strategy whatsoever, and match it up against any other strategy to see how it will perform.

Benefit ordering Once the tournament has been played out, we can take the average benefit of each player strategy (gene) in the population. We can then rank these against all the other strategies in the population to see which performed the best. A chosen number k of the top performing strategies are kept for the following generation. They are deemed to have *survived* within that generation.

Defining parents Based on the performance of each strategy and their ranking against all others' benefit ordering in the population, each strategy is given a weighting in order to represent the probability of them being chosen to be parents and reproduce offspring for the next generation (even if they haven't been chosen to survive themselves). For the method of weighting we use the Logit best response dynamics such that the weighting w_i for strategy i is

$$w_i = \frac{\exp \beta(s_i)}{\sum_{j=1}^{N} \exp \beta(s_j)}$$

where $\beta(s_i)$ is the payoff of strategy i. This ensures each weighting $0 < w_i < 1$ and $\sum_{i=1}^{n} w_i = 1$. It can also incorporate negative payoffs, which is useful since not all payoffs in the tournaments will be positive. In fact, in some cases there might only be a handful that are positive in a generation. Two lists of parents will be chosen, aptly named *mothers* and *fathers*. Therefore, if we have a population of size N and decide to keep k genes (strategies) through to the next generation then we will choose at random (with replacement) two lists of $N-k$ parents.

Producing offspring The algorithm then aligns the two lists of parents (mothers and fathers) and *mates* them, iterating through each resource they have a strategy for and, at the flip of a coin, choosing which strategy to take. Therefore, each strategy on a resource has a 50% chance of being chosen. For example, if we have a mother $m = (\boldsymbol{s}^m, \boldsymbol{\alpha}^m)$ and father $f = (\boldsymbol{s}^f, \boldsymbol{\alpha}^f)$, we iterate through each resource the game is played over. The *child* gene c created will have a mixture of properties from the mother and father such that (for example)

$$c = (\boldsymbol{s}^c, \boldsymbol{\alpha}^c) = ((s_0^m, s_1^f, s_2^f, \ldots, s_{n-1}^m, s_n^f), (\alpha_0^m, \alpha_1^f, \alpha_2^f, \ldots, \alpha_{n-1}^m, \alpha_n^f))$$

Noise mutations Once the offspring have been produced we have N genes for the next population, we apply some noise to the strategies. This is to ensure there is some deviation and

movement from generation to generation and the algorithm does not find a suboptimal solution and settle on it too fast. However, eventually we do require the algorithm to settle once it has found the true (hopefully unique) optimal solution. We therefore introduce a function of diminishing noise, that decreases as the number of generations increase. Denote $0 < \epsilon < 1$ to be a bound for the noise. A value m is chosen uniformly at random such that $m \in [-\epsilon, \epsilon]$. α is then mutated to a new rate

$$\alpha \cdot (1 + m/\ln(gen)).$$

where we define gen to be the current generation.

Full mutations After each generation, the algorithm also, with some probability parameter, introduces a full mutation, or immigrant into the population. This means a randomly generated strategy within the population's possible strategy classes. Again, this is to ensure the algorithm doesn't settle too quickly on a suboptimal solution. We define a mutation rate η to be the probability of an immigrant being introduced into the population.

Deterministic and stochastic simulation We consider two types of simulation, *deterministic* and *stochastic*. In deterministic simulation, the player benefits are returned by an analytic formula (modelled numerically). This is the case for strategy classes we have analysed mathematically such as periodic and non-arithmetic renewal strategy classes. In stochastic simulation, each round is run over a fixed time span within the python framework we have designed. This means that provided a strategy class can be coded algorithmically, it can be simulated. Therefore, stochastic simulation means a much broader class of strategies can be tested.

6.2 Genetic learning in the game of exponential multi-rate threshold FlipThem

Now we have defined our genetic algorithm, we can begin to experiment with the success of convergence to our analytic equilibria calculated in previous chapters. We show that, for certain cases, our genetic algorithm optimises both players' populations towards the equilibrium that we have calculated in Chapter 4. We discuss these cases, and others that do not converge to calculated equilibria.

6.2.1 Exponential FlipIt

We begin with the simplest case: exponential FlipIt. This is played over one resource and both players play with rates sampled from an exponential distribution. We have already fully analysed this case mathematically so know the payoffs for each player with respect to their costs and rates of play. Therefore, we first use deterministic simulation and then follow with stochastic

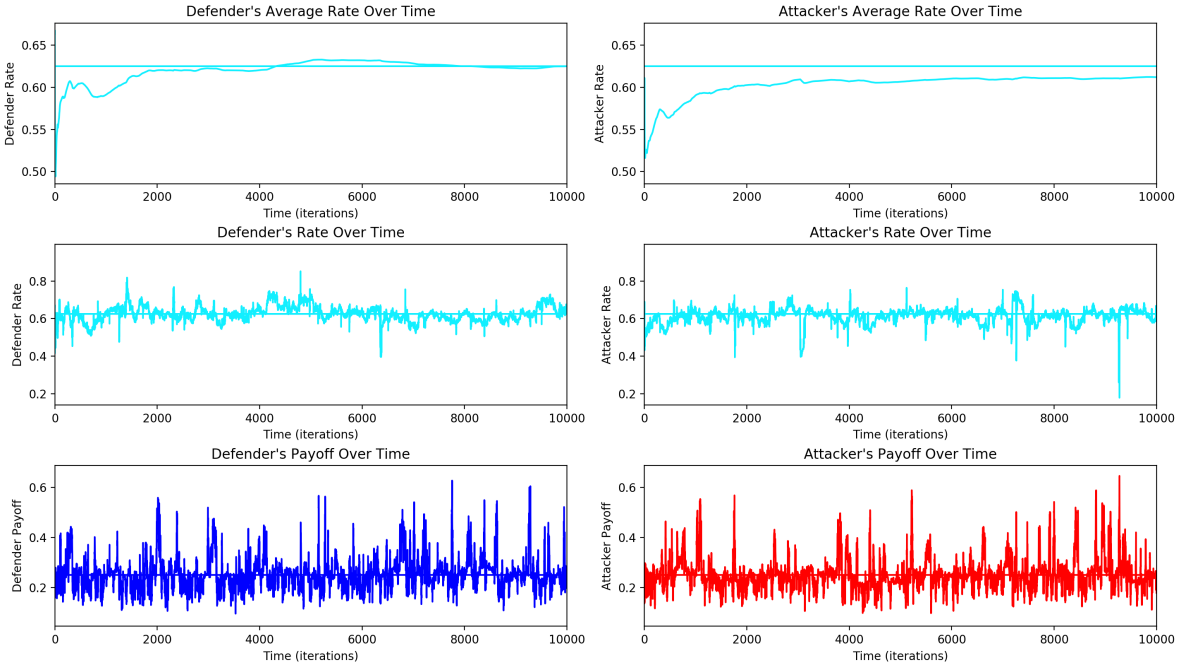


Figure 6.1: The evolution of two populations representing defender and attacker strategies sampled from the exponential strategy class for up to 10000 iterations.

simulation. As a reminder from Section 4.3 these are

$$\beta_D(\mu,\lambda) = 1 - \frac{\lambda}{\mu+\lambda} - c^D\mu, \qquad \beta_A(\mu,\lambda) = \frac{\lambda}{\mu+\lambda} - c^A\lambda \tag{6.1}$$

where μ and λ are the exponential rates of play of the defender and attacker, respectively. We assign costs for both players to be $c^D = c^A = 0.4$. We expect player populations to converge to

$$\mu^* = \frac{c^D}{(c^A + c^D)^2} \quad \text{and} \quad \lambda^* = \frac{c^A}{(c^A + c^D)^2} \tag{6.2}$$

for the defender and attacker's populations, respectively. The player costs are identical meaning the game is symmetric. Therefore, we expect both populations to converge to the same rate of 0.625. The graphs on the second row of Figure 6.1 represent the average of the top ranked strategies that are kept into the following generation. We can see that these values hover around the horizontal lines representing equilibrium. The graphs on the row above are these rates averaged over time. We see that as time increases, the averages of the players' rates approach the same equilibrium. On the final row of Figure 6.1, we have the averages of the payoffs of the strategies kept through to the next generation. We can see that whilst the values hover around the horizontal line that represents the payoffs received if both players were to play exactly at their respective equilibrium rates, there is a lot of movement. There are also a lot of sharp drops and rises from both players with respect to the payoffs.

In Figure 6.2, we have zoomed in to explore the behaviour of the erratic payoffs. The graphs are plotted between iterations 8000 and 10000. We can see that the defender's average rate

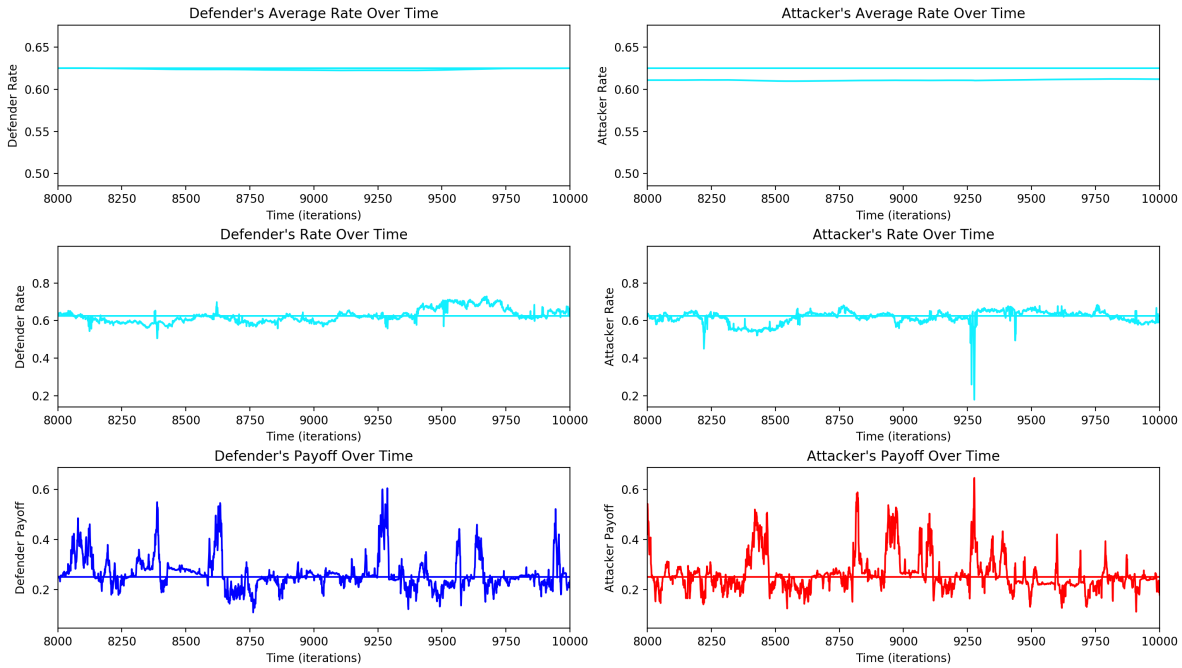


Figure 6.2: The evolution of two populations representing defender and attacker strategies sampled from the exponential strategy class from iteration 8000 to 10000.

over time is on the equilibrium line and the attacker's very close to it. Likewise, on the second row both defender's and attacker's specific 'kept' rates are very close to the line. However, at around iteration 9250 the attacker's rate drops sharply. We can see that with this there is a sharp increase in payoff for the attacker. In Figure 6.3, we have zoomed in again to between iterations 9200 and 9300. We can see that it's actually the defender's payoff that rises first at just before iteration 9250. There is a slight decrease in the 'kept' defender rates, however it seems unlikely that such a small change in rates would account for such a large increase in payoff. It would assume that the defender receives the same amount of time in control of the resource whilst lowering her move rate and hence less cost for moving.

After a short decrease in payoff for the attacker, his payoff starts to increase quite dramatically. This is coupled with a sharp decrease in 'kept' attacker rates. Again, this could potentially account for the attacker's payoff increase, and it is more likely since there is a larger decrease than compared to the defender's. Notice however, that the defender's payoff also stays above the equilibrium line for the whole time the attacker's does. We think this is be due to the fact that the tournament is only sampling 30% of all the available matches to be played. Therefore, this could be an occurrence of where a suboptimal strategy only plays against other suboptimal strategies and therefore produces stronger results than if the strategy played against more optimal strategies. This means that for a short amount of time this strategy would skew the results, looking a better prospect than it actually is. We can see that after this short period, the lower rate strategy gets dominated and the graph moves back towards the equilibrium line.

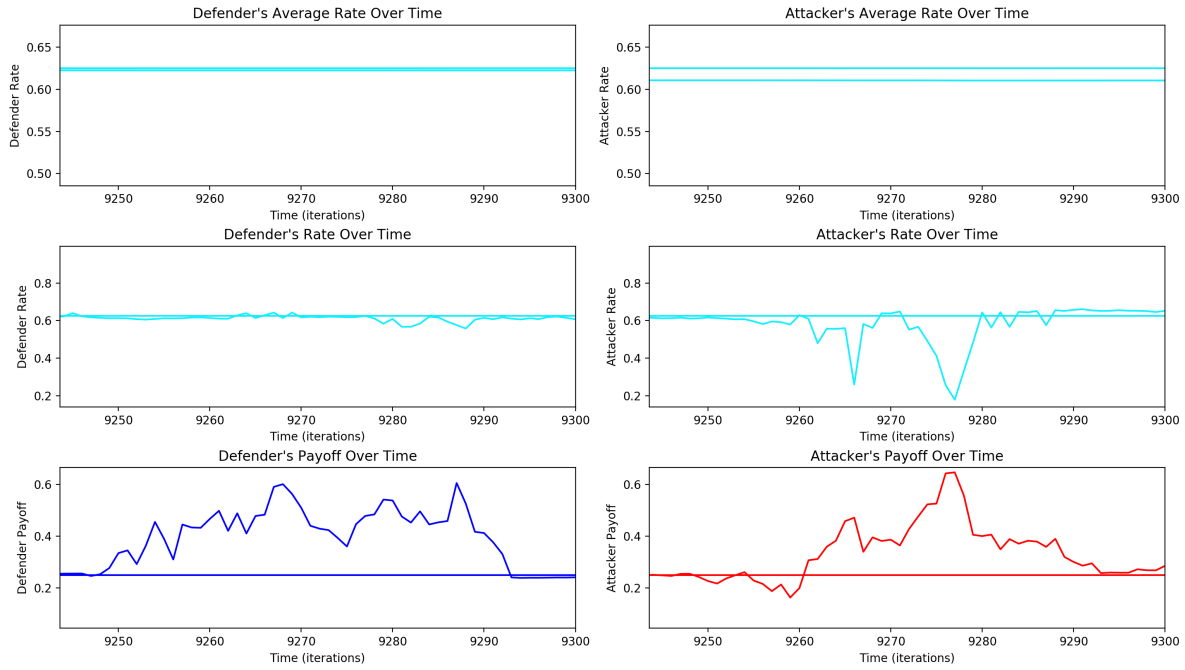


Figure 6.3: The evolution of two populations representing defender and attacker strategies sampled from the exponential strategy class from iteration 9200 to 9300.

Given more time, one could record more detailed statistics on each strategy's opponents to draw up a better picture on how the performance of populations varies from generation to generation.

Playing stochastically As explained above, our main goal is to be able to simulate these results and have them converge to our calculated equilibrium for strategy classes that we have researched analytically. This gives us confidence when we start exploring the unknown strategy classes. In Figure 6.4, we can see that we get similar results to the deterministic case. Thus, we start to get a good idea that our simulations are behaving nicely. In fact, providing simulation based data seems to have provided some calm within the chaos. There is order to the randomness, meaning the rates hover evenly around the analytic equilibria. This makes sense since the simulations are more stochastic than the strict reward functions and thus we have two forms of noise.

3 resource partial threshold Now we have convergence in the FlipIt case, the obvious extension is to consider the multi-rate partial threshold case. For this, we look at multi-rate $(3,2)$-FlipThem with defender costs of $(0.3, 0.38, 0.45)$ and attacker costs of $(0.3, 0.32, 0.36)$. Figure 6.5 shows the results of the simulated genetic algorithm. Once again, we see convergence. We find that this is the case for any number of resources and threshold; we have a genetic algorithm that converges to the calculated equilibrium for any exponential threshold FlipThem case.

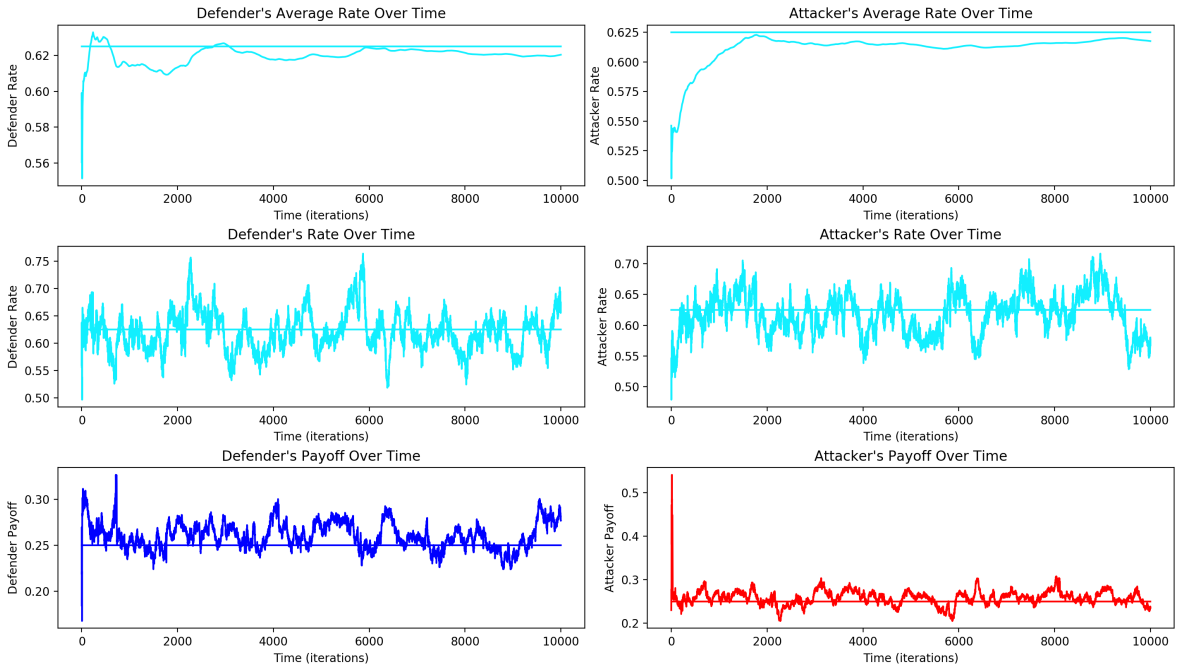


Figure 6.4: The simulated evolution of two populations representing defender and attacker strategies in the game of exponential FlipIt for up to 10000 iterations.

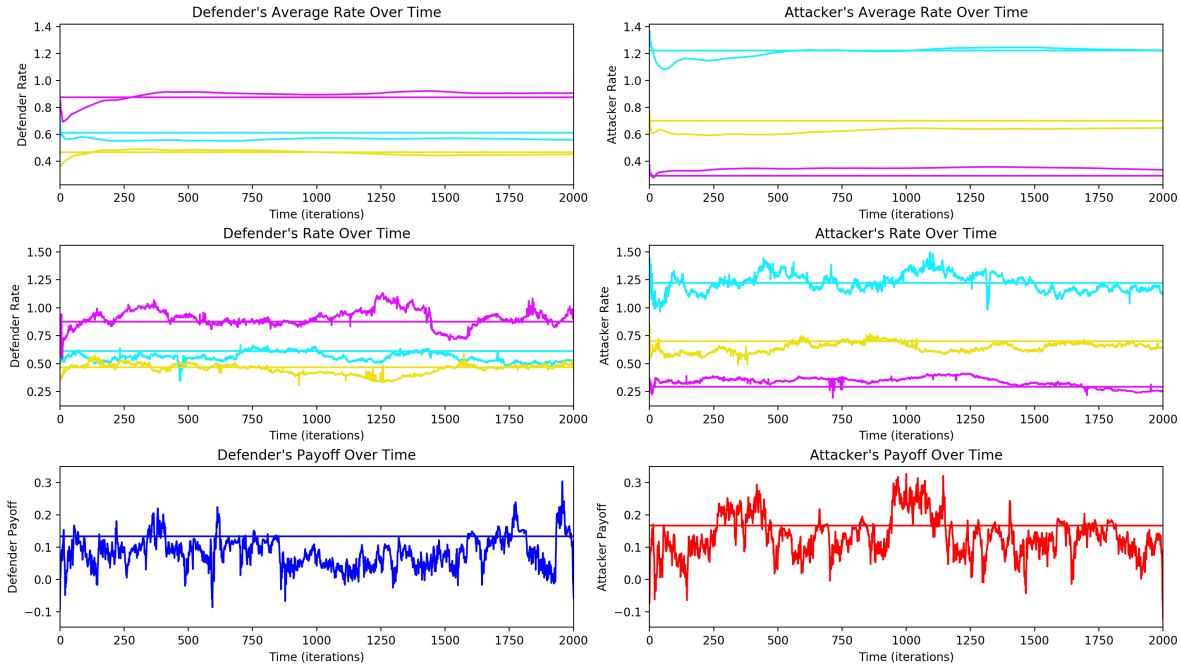


Figure 6.5: The simulated evolution of two populations representing defender and attacker strategies in the game of exponential multi-rate $(3,2)$-FlipThem for up to 2000 iterations.

6.3 Genetic learning in the game of periodic multi-rate threshold FlipThem

In this section, we move onto the periodic strategy space. For the exponential class above we have analysed it fully, finding the Nash equilibria for all cases. However, in the periodic case we have not. Therefore, using genetic algorithms to learn best responses and equilibria is an extremely useful tool for analysing various strategy classes that are difficult to obtain mathematically. The only case we have calculated the equilibrium for is the single resource FlipIt game discussed in Section 4.1. Therefore, we begin there. We find that optimising player rates is not as easy as with exponential.

6.3.1 Finding best responses

A useful feature of the genetic algorithm approach is we can find optimal responses to any rate of play used by the opponent. Therefore, we first fix the attacker to play at a certain rate away from the equilibrium and see if the defender can optimise her strategy (whilst still within the periodic space) in order to maximise her payoff.

Periodic FlipIt Despite being (conceptually) the simplest case in this work, periodic FlipIt game has a lot of mileage in terms of analysis and also simulation which we discuss here. A reminder of the payoff functions from Section 4.1 for both players playing periodic strategies with rates α^D and α^A for the defender and attacker, respectively gives:
Case 1: $\alpha^D \geq \alpha^A$ (The defender plays faster or equal speed to the attacker)

$$\beta^D(\alpha^D,\alpha^A) = 1 - \frac{\alpha^A}{2\cdot\alpha^D} - c^D\cdot\alpha^D,$$
$$\beta^A(\alpha^D,\alpha^A) = \frac{\alpha^A}{2\cdot\alpha^D} - c^A\cdot\alpha^A.$$

Case 2: $\alpha^D < \alpha^A$ (The defender plays slower than the attacker)

$$\beta^D(\alpha^D,\alpha^A) = \frac{\alpha^D}{2\cdot\alpha^A} - c^D\cdot\alpha^D,$$
$$\beta^A(\alpha^D,\alpha^A) = 1 - \frac{\alpha^D}{2\cdot\alpha^A} - c^A\cdot\alpha^A.$$

In [27] they find best responses, so that for the defender the best response function is

$$\text{br}^D(\alpha^A) = \begin{cases} \sqrt{\frac{\alpha^A}{2\cdot c^D}} & \text{if } \alpha^A < \frac{1}{2\cdot c^D}, \\ 0 \quad \text{if } \alpha^A > \frac{1}{2\cdot c^D}, \\ [0, \sqrt{\frac{\alpha^A}{2\cdot c^D}}], \quad \text{if } \alpha^A = \frac{1}{2\cdot c^D}. \end{cases} \tag{6.3}$$

and a similar result can be found for the attacker's best response based on the defender's rate of play. Thus, we consider three separate cases based on the attacker's rate in relation to the

defender's move cost. Fix the defender's cost to be $c^D = 0.2$. For the first case, shown in Figure 6.6 we fix the attacker's rate to be $\alpha^A = 0.8$, so that $\alpha^A < \frac{1}{2 \cdot c^D} = 2.5$. We can see that the defender's rate converges to the desired rate of approximately 1.41.

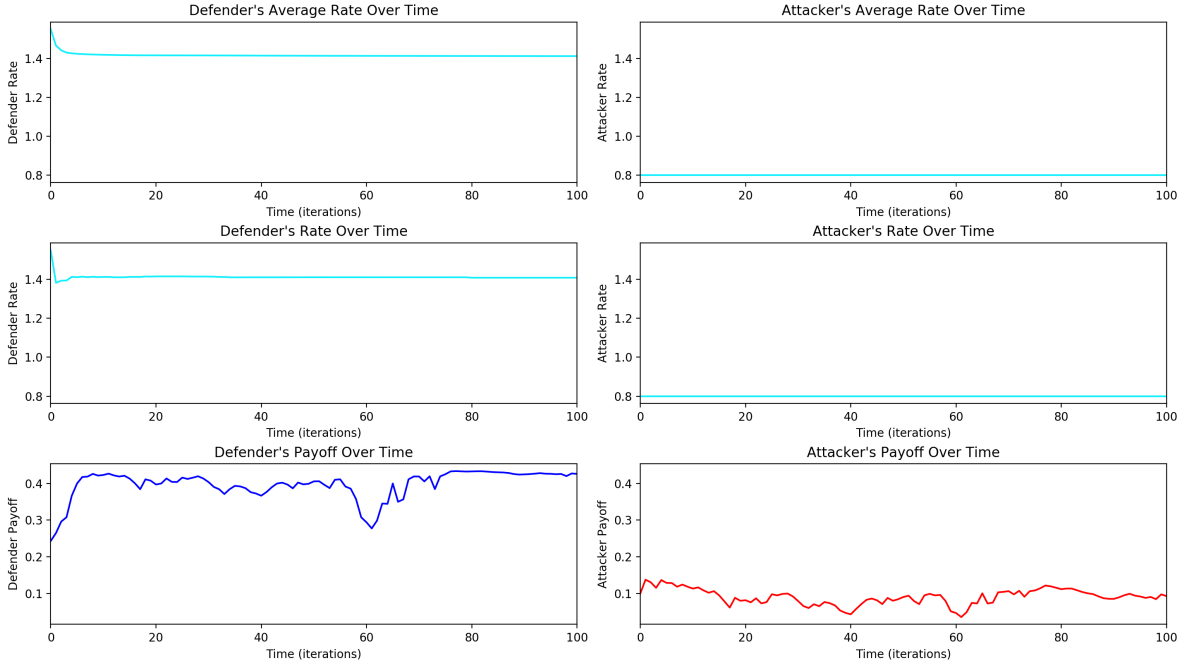


Figure 6.6: A defender population against a fixed single attacker playing an periodic strategy with rate 0.8. Move costs are 0.2 for both defender and attacker

For the second case, in which $\alpha^A > \frac{1}{2 \cdot c^D} = 2.5$, we fix $\alpha^A = 2.6$ so that the defender's rate should approach zero. This case is shown in Figure 6.7. We see that in the "defender's rate over time" graph on the second row of Figure 6.7 the rate approaches zero very fast, confirming our analysis.

Finally, we look at the third case, such that $\alpha^A = \frac{1}{2 \cdot c^D} = \frac{1}{2 \cdot 0.2} = 2.5$. This case is shown in Figure 6.8. The best response is an interval, therefore we are unsure if the genetic algorithm will settle on a specific rate. However, we see that it not only settles but seems to converge to a point above the calculated best response interval. Figure 6.9 shows a plot of the defender's and attacker's rewards with respect to a fixed attacker's rate of 2.5 and a varied defender's rate. We can see that with the defender deciding to drop out of the game she receives zero reward, as expected. However, even if the defender decides to enter the game there is no incentive since she will still receive zero benefit. As the defender increases her rate, the attacker's benefit decreases. Once the defender's rate is over 2.5, both her's and the attacker's benefits go below zero. This shows that it will be extremely hard, if not impossible, for the defender's population to converge upon something meaningful since there is little to no difference between playing at close to zero or close to 2.5 and everything in between.

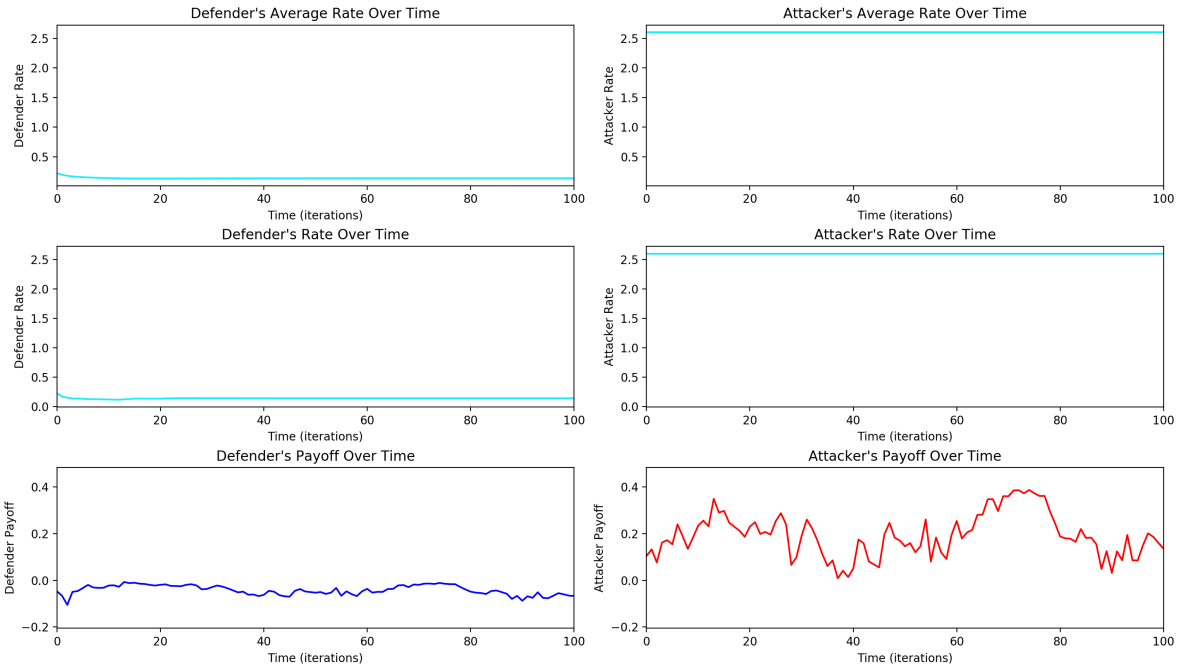


Figure 6.7: A defender population against a fixed single attacker playing a periodic strategy with rate 0.8. Move costs are 0.2 for both defender and attacker.

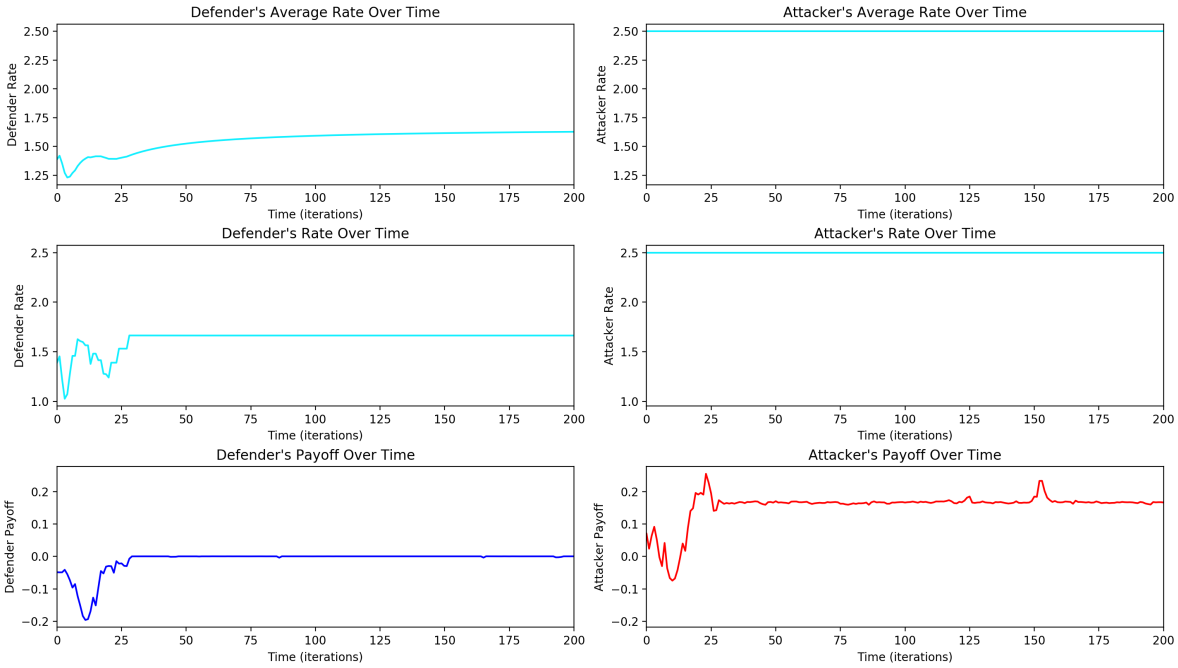


Figure 6.8: A defender population against a fixed single attacker playing an periodic strategy with rate 2.5. Move costs are 0.2 for both defender and attacker

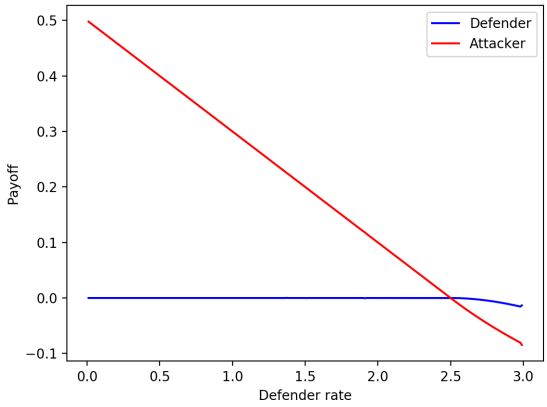


Figure 6.9: Fixed periodic attacker's rate of 2.5 and varied the periodic defender's rate. Defender's payoff is the blue line, attacker's payoff is the orange line.

6.3.2 Finding equilibria, fixing one population

In the section above, we fixed the attacker's population at a suboptimal point in the strategy space. In this section, we fix one of the player's rates to play at the equilibrium based on move costs and see whether the genetic algorithm settles on the other player's equilibrium rate. This is an important step towards our genetic algorithm learning the best rates to play within the strategy class. We first remind ourselves of the analytic equilibria from Section 4.1 and Theorem 4.2. The Nash equilibria are

$$\begin{aligned} c^D < c^A \quad & \alpha^{D*} = \frac{1}{2 \cdot c^A}, \quad \alpha^{A*} = \frac{c^D}{2 \cdot (c^A)^2}, \\ c^D > c^A \quad & \alpha^{D*} = \frac{c^A}{2 \cdot (c^D)^2}, \quad \alpha^{A*} = \frac{1}{2 \cdot c^D}, \\ c^D = c^A \quad & \alpha^{D*} = \alpha^{A*} = \frac{1}{2 \cdot c^D}. \end{aligned}$$

Therefore, once again we have three cases to consider. We fix the attacker's rates again in all examples noting that the game is symmetric. For the first case, we fix move costs at $c^D = 0.2, c^A = 0.3$ for the defender and attacker, respectively. Therefore, at equilibrium the attacker's rate is $\alpha^{A*} = \frac{c^D}{2 \cdot (c^A)^2} \approx 1.11$. We expect the defender's rate to settle on $\alpha^{D*} = \frac{1}{2 \cdot c^A} \approx 1.67$. We see this is the case in Figure 6.10, with the defender's rate converging quickly to the desired value.

For the second case, we fix move costs at $c^D = 0.3, c^A = 0.2$ for the defender and attacker, respectively. In other words, the reverse of case 1. Therefore, at equilibrium the attacker's rate is $\alpha^{A*} = \frac{1}{2 \cdot c^D} \approx 1.667$. We expect the defender's rate to settle on $\alpha^{D*} = \frac{c^A}{2 \cdot (c^D)^2} \approx 1.11$. We do not see this is the case in Figure 6.11.

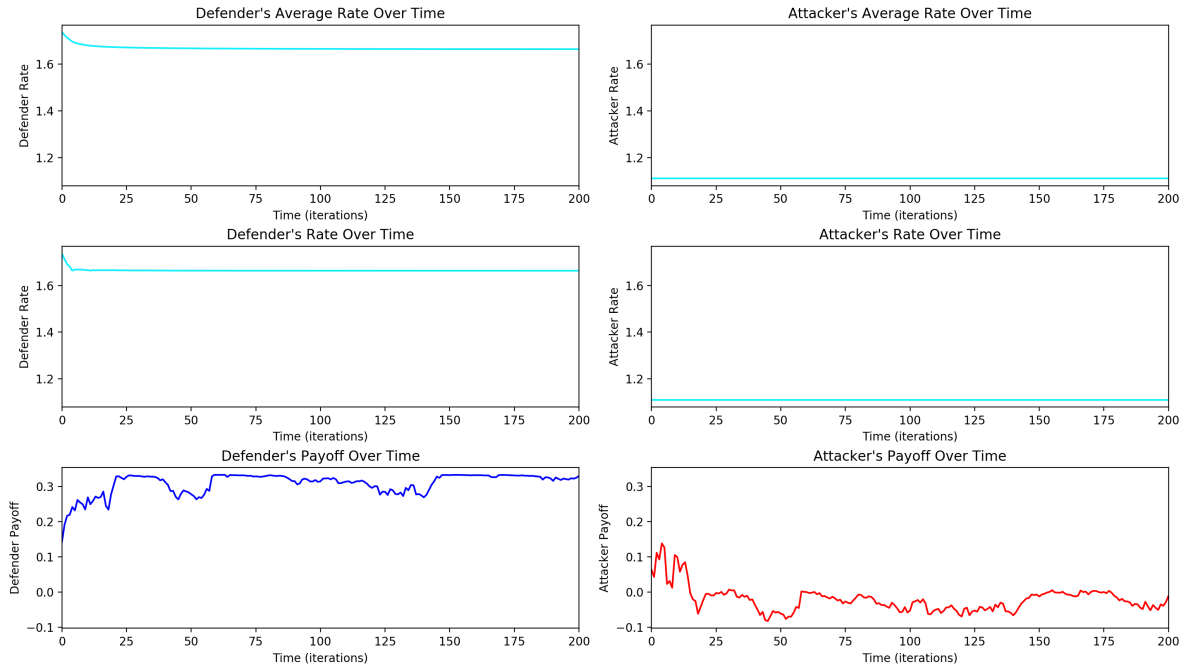


Figure 6.10: A defender population against a fixed single attacker playing an periodic strategy at equilibrium with rate 1.11. Move costs are 0.2 and 0.3 for the defender and attacker, respectively.

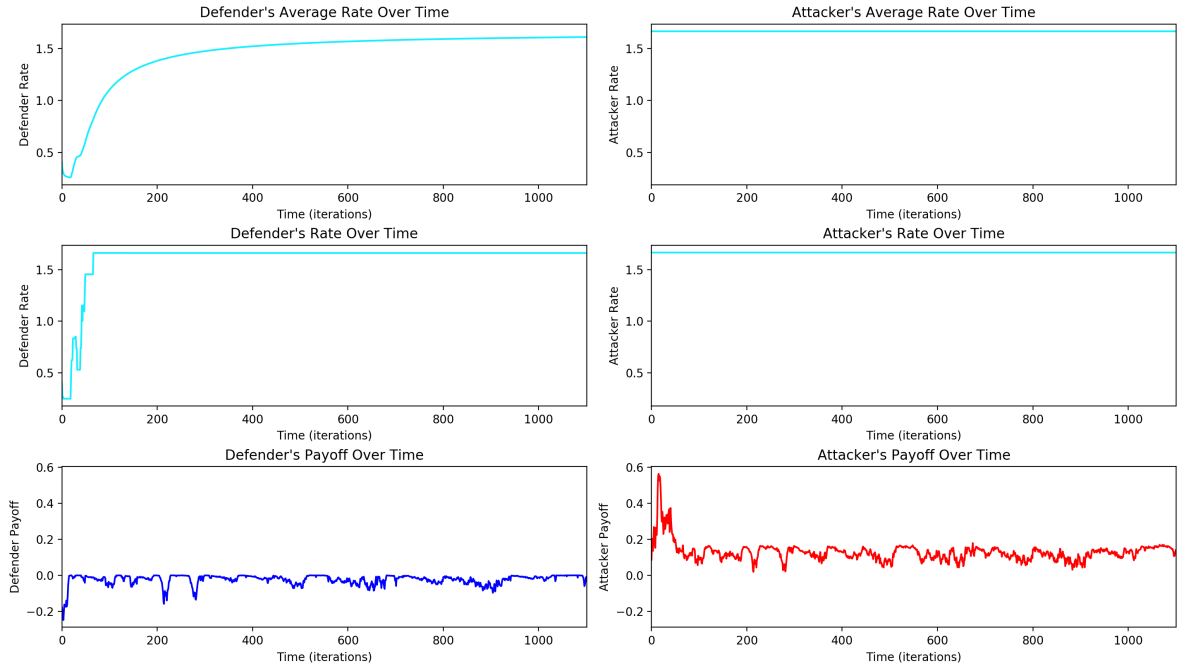


Figure 6.11: A defender population against a fixed single attacker playing an periodic strategy at equilibrium with rate 1.667. Move costs are 0.3 and 0.2 for the defender and attacker, respectively.

For the third case, we fix move costs at $c^D = c^A = 0.3$ for the defender and attacker, respectively. Therefore, at equilibrium the attacker's rate is $\alpha^{A*} = \frac{c^D}{2\cdot(c^A)^2} \approx 1.667$. We expect the defender's rate to settle on $\alpha^{D*} = \frac{1}{2\cdot c^A} \approx 1.667$. We do not see this convergence in Figure 6.12. We believe the issues with convergence is due to similar interval issues as from the best response equation (6.3). A player's rate can settle within an interval and hence has no incentive to change their strategy. Therefore, (theoretically) weaker strategies can survive within the population and even thrive, producing results that do not converge to the calculated equilibrium. This could perhaps be solved by adding more noise to the mutations of the genetic algorithm in order to 'knock' the population off a specific convergence path.

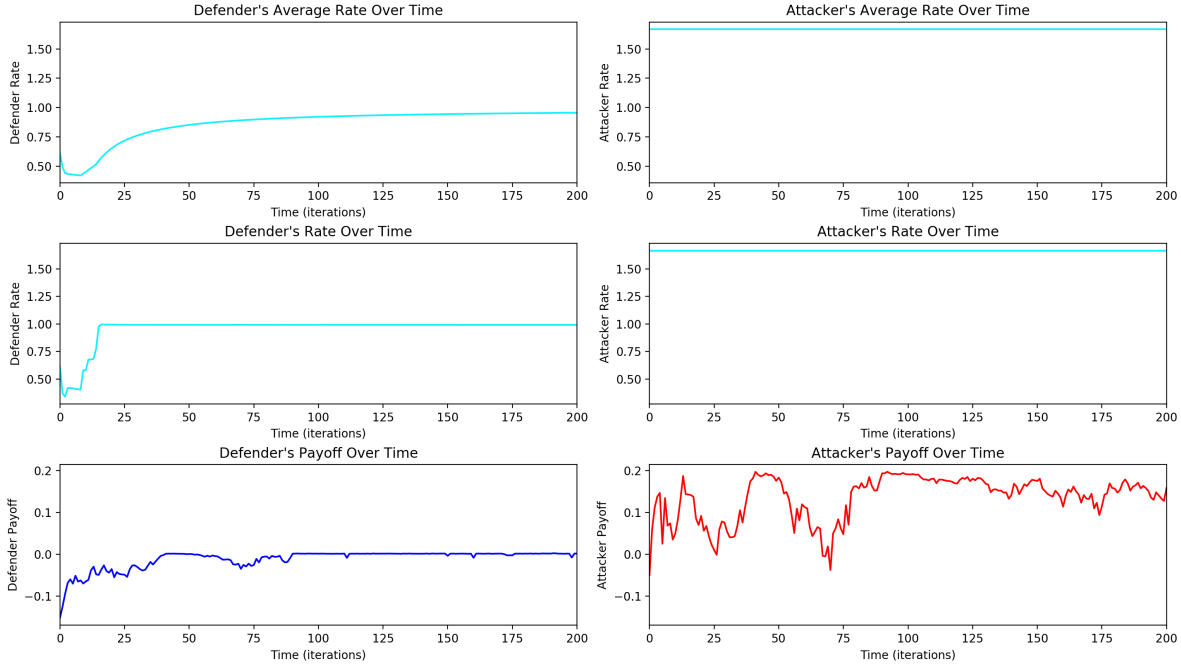


Figure 6.12: A defender population against a fixed single attacker playing a periodic strategy at equilibrium with rate 1.667. Move costs are 0.3 for both the defender and attacker.

6.3.3 Finding equilibria, evolving both populations

Finally, we allow both populations to evolve. Each population has 50 genes consisting of strategies from the periodic strategy class. Once again, we have three cases to consider depending on the player move costs, or rather, ratios between them. However, since both populations are now identical in set-up, we now only have to consider the cases when players' move costs are equal or unequal. In Figure 6.13 the defender's move cost is 0.2 and attacker's is 0.3 so that the defender's equilibrium rate of play is approximately 1.67 and attacker's equilibrium rate of play is approximately 1.11. However, we see in Figure 6.13 the defender's rate struggles to settle on any specific value. Likewise, the attacker's rate doesn't settle anywhere specific either. Again, in Figure 6.14, where the move costs are equal at 0.3 neither populations converge to a

meaningful value. We believe this is because of the reasons above in the fixed population case; both the defender and attacker have no incentive to always attempt to play at a strict equilibrium rate. There is a lot more noise in the movement of the rates and the benefit functions. This is due to varying both populations simultaneously and potentially due to the sampling rate. Once again, a certain strategy might only get to play the opponent population's weaker strategies and therefore receives a better average reward than other stronger strategies in their population. Obviously, a suggested solution to this is to increase the sampling rate. In order to do this, one would need to either improve the efficiency of the code, run of a more powerful computer, or both.

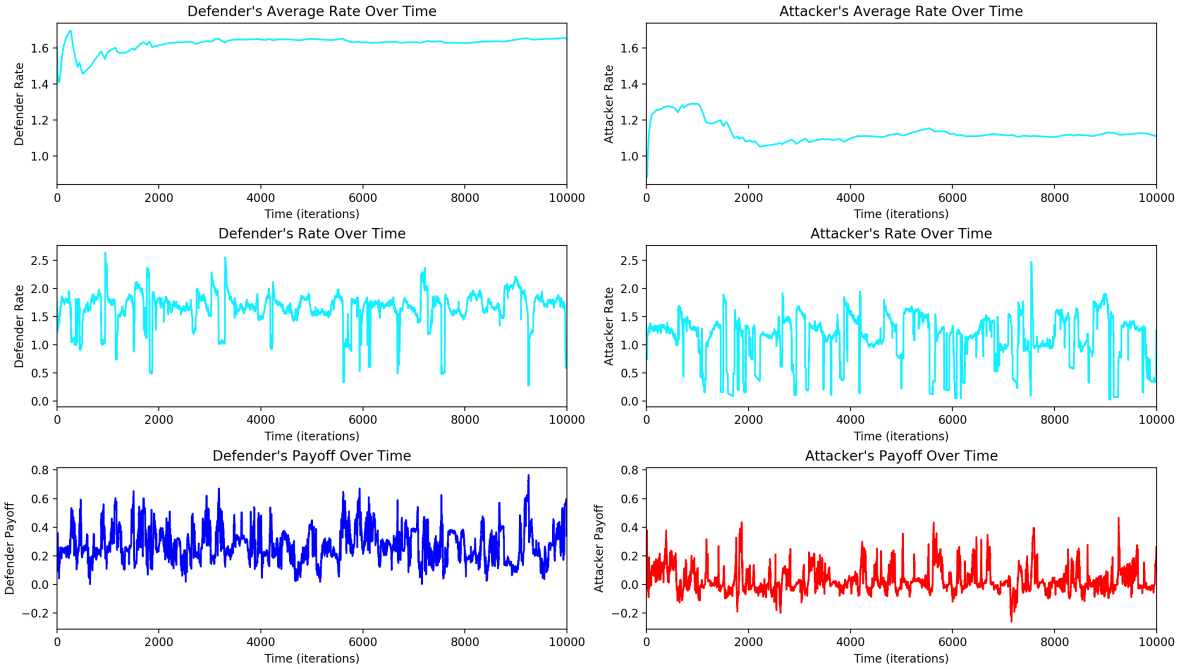


Figure 6.13: A periodic defender population against a periodic attacker playing a periodic strategy. Move costs are 0.2 and 0.3 for the defender and attacker, respectively.

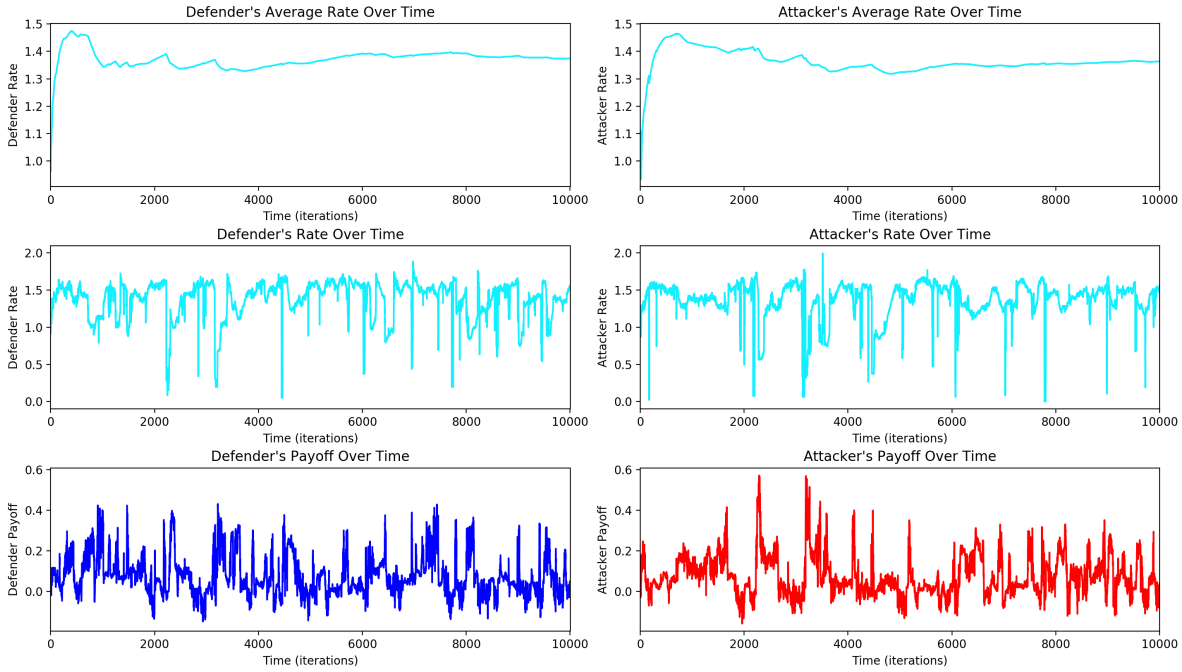


Figure 6.14: A periodic defender population against a periodic attacker playing a periodic strategy. Move costs are 0.3 for both the defender and attacker.

6.4 Evolving multiple strategy class populations

In this section, we explore multiple class strategy spaces. By this, we mean a population that consists of more than one class of strategy. For example, a population could contain a proportion of periodic and exponential strategies. When playing over multiple resources, there can be multiple strategy classes within one gene, playing different strategy classes over different resources. This is difficult to analyse mathematically, therefore our genetic algorithm is a very useful tool to explore multiple class strategy spaces. Since we have two populations evolving simultaneously, some strategies will be the dominant in their population when playing only very specific strategies in the opposing population. This will become clearer over time.

6.4.1 Non-adaptive strategies

The first strategy classes we look at are the non-adaptive strategies. Namely, the periodic and exponential strategy classes. In [27], van Dijk et al. prove the dominant strategy class within the class of general renewal strategies to be periodic. We begin by seeing if our genetic algorithm agrees with these results.

Just how strong is the periodic strategy? In the original FlipIt paper [27], the authors show mathematically that in the non-adaptive game the periodic strategy class dominates all other non-arithmetic renewal strategies, in particular over exponential strategies. We can now

confirm this with our genetic algorithm. In Figure 6.15 the defender population consists of periodic and exponential strategies. The attacker population consists of only periodic strategies. This means the genetic algorithm will most likely evolve the defender population in such a way that it competes best against periodic strategies. In Figure 6.15 we see for the majority of iterations the defender population is dominated by periodic strategies. This agrees with the analysis conducted in [27]. Through the whole simulation we see slight perturbations in the strategy counts. This is due to a non-zero mutation rate that keeps introducing random strategies into the population (from within the periodic and exponential classes). This is to stop the population from converging upon a suboptimal strategy. Between approximately iteration 200 and 300 we see a larger perturbation within the strategy count. We believe this is because while the periodic strategy is clearly the more dominant strategy at a specific rate, a good rate exponential strategy will still trump a bad rate periodic strategy.

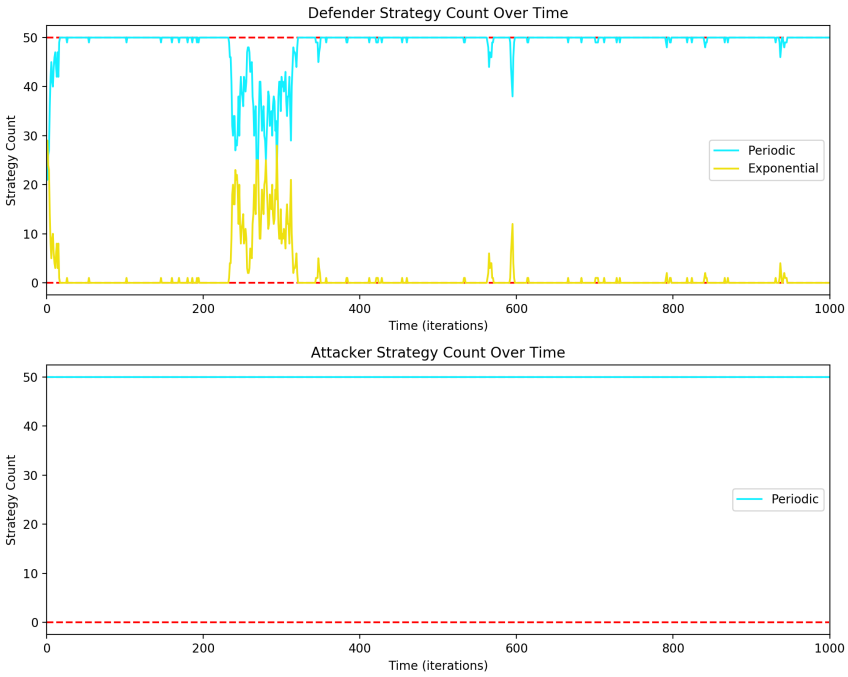


Figure 6.15: Periodic/exponential defender population against a periodic attacker population

In Figure 6.16, the defender population consists of periodic and exponential strategies again. However, this time the attacker population consists of only exponential strategies. Therefore, this time we expect the genetic algorithm to evolve the defender population in such a way that it competes best against exponential strategies. In Figure 6.15, we see for the majority of iterations the defender population is dominated by periodic strategies. Once again, this agrees with the analysis conducted in [27].

Finally, for the non-adaptive classes we consider both populations having periodic and ex-

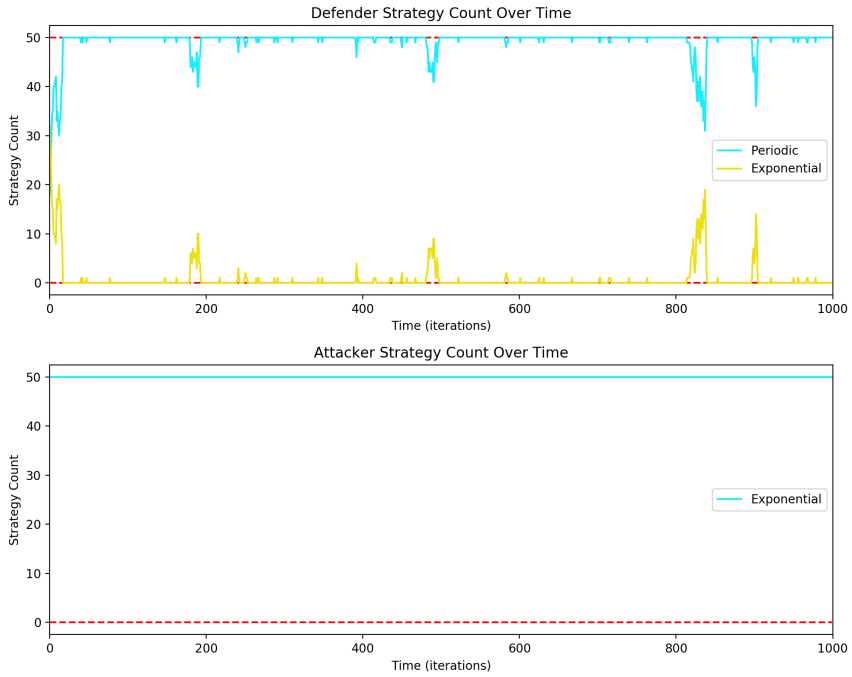


Figure 6.16: Periodic/exponential defender population against an exponential attacker population

ponential strategies. So far, the genetic algorithm has shown the periodic strategy to be the dominant strategy over the exponential when playing against both periodic strategies and exponential strategies. Therefore, we expect the periodic strategy to dominate in both attacker and defender populations. We see this is the case in Figure 6.17, with periodic coming out on top for the majority of iterations.

6.4.2 Adaptive strategy classes

In this section, we consider how adaptive strategy classes perform against non-adaptive strategy classes. In adaptive strategies, players receive information about their opponent's moves after they have moved themselves. We consider the player to have received *feedback*. The specific strategy class we consider is named the *last move* strategy class, which we introduce here.

6.4.2.1 The last move strategy Following notation defined in Chapter 2, we denote ϕ_r^i to be the feedback received by Player i when they move on resource $\mathscr{R}_r$. If player i moves on resource $\mathscr{R}_r$ at time t, the feedback given to that player will be

$$\phi_r^i(t) = t_{r,j}^{-i} \tag{6.4}$$

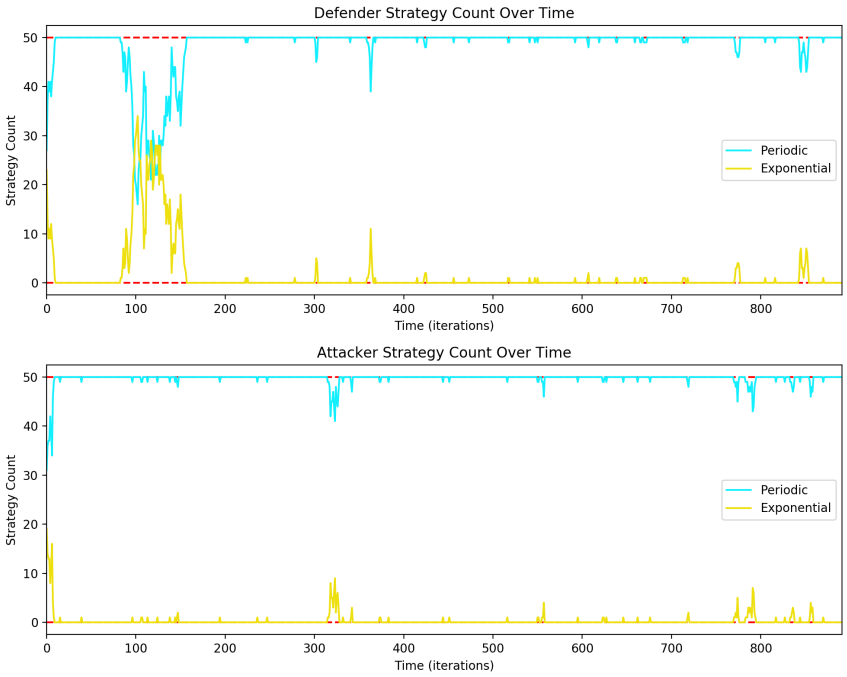


Figure 6.17: Periodic/exponential defender population against a periodic/exponential attacker population

where $t^{-i}_{r,j} < t$ is player $-i$'s most recent move time on resource $\mathscr{R}_r$. Therefore, if player i has just made their mth move, the player's view will be

$$\mathbf{v}^i_r(t_m) = ((t^i_{r,1},\ \phi^i_r(t^i_{r,1})),\ (t^i_{r,2},\ \phi^i_r(t^i_{r,2})),\ \ldots,\ (t^i_{r,m},\ \phi^i_r(t^i_{r,m})))$$

where $\phi^i_r(t^i_{r,j})$ is defined in 6.4. As the game goes on, the last move player will build up a view of when the opponent moves and try to react accordingly. More formally, the last move player starts off as a periodic (with random phase) strategy until they form a significant enough view to start adapting to their opponent's strategy. We deem *significant enough* in this case to be two moves.

From then on, the last move player calculates their belief of the opponent's move rate $\tilde{\alpha}^{-i}_r$ by

$$\tilde{\alpha}^{-i}_r = \phi^i_r\left(t^i_{r,j}\right) - \phi^i_r\left(t^i_{r,j-1}\right).$$

This means they look at the player's last two observed moves of the opponent and assumes the periodic rate they are playing is simply the difference between the two move times. The last move player calculates their next move time on resource $\mathscr{R}_r$ by

$$\phi^i_r\left(t^i_{r,j}\right) + \tilde{\alpha}^{-i}_r + \epsilon = 2 \cdot \phi^i_r\left(t^i_{r,j}\right) - \phi^i_r\left(t^i_{r,j-1}\right) + \epsilon$$

where ϵ is a number close to zero. The keen eyed reader should notice an important caveat to these formulae. If the last move player were to move quicker than their opponent then they would

receive feedback on exactly the same move previously observed, thus creating a zero rate belief. If this occurs, we assume the player will move another step back in their view such that their belief of their opponents rate is no longer zero. Since we are doing this algorithmically, this is a very easy process.

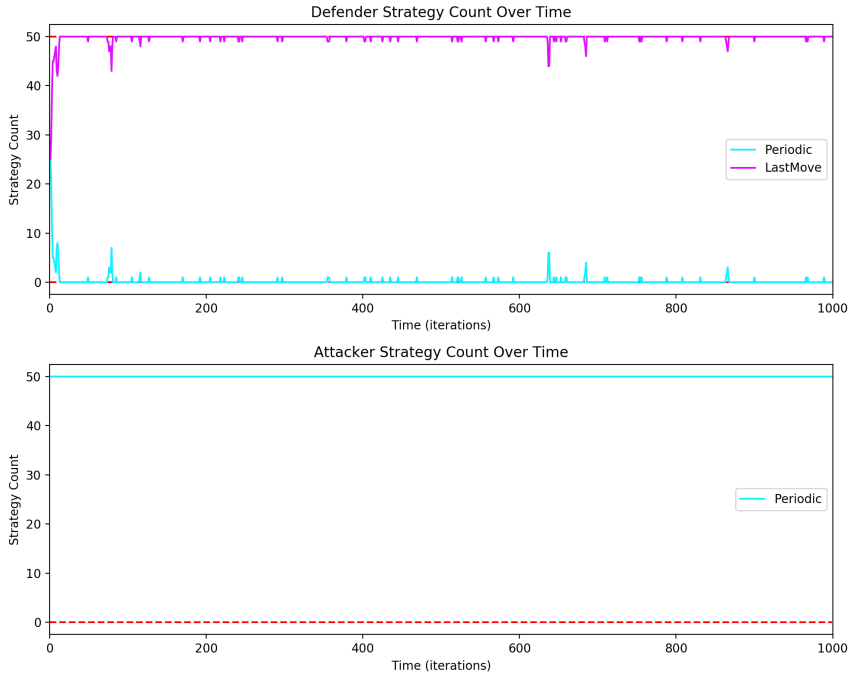


Figure 6.18: Periodic/last move defender population against a periodic attacker population

Last move against non-adaptive strategies The first example we consider is a periodic/last move defender population against periodic attacker population shown in Figure 6.18. Since within the last move strategy class it assumes their opponent is playing a periodic strategy and adapts to the information it receives, we expect a strong performance from the last move strategy class. We can see this in Figure 6.18, the last move strategy completely dominates the periodic strategy class.

In Figure 6.19, we consider how strong the last move strategy is compared to the periodic strategy when playing against an exponential attacker. We know the last move strategy is tailored towards a periodic strategy opponent and therefore we might expect poorer performance against other strategy classes. Indeed this is the case when playing the exponential strategies. Since the last move strategy uses the previously observed move times of the opponent in order to calculate their belief of their opponents rates, when playing the exponential strategy the last two moves of an exponential strategy class give no information regarding the next moves and in particular the rate of play. Therefore, the last move strategy struggles, as shown in Figure 6.19.

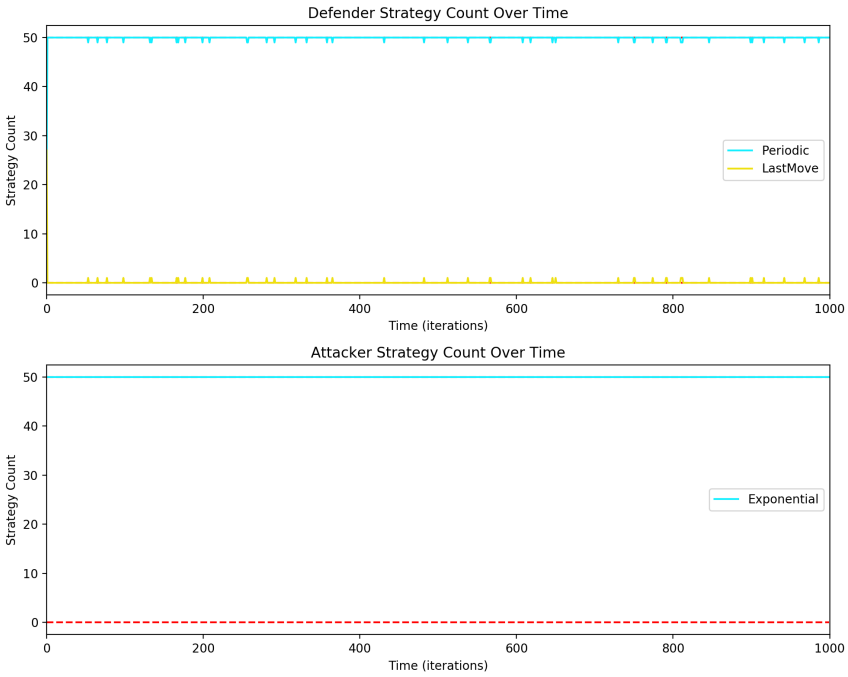


Figure 6.19: Periodic/last move defender population against an exponential attacker population

Next, we consider which is the stronger strategy between periodic and last move against an attacker population containing both periodic and exponential strategies. In Figure 6.20, we see a cyclic result. Speaking from the defender's perspective, we know that last move is strongest when the attacker is playing a periodic strategy, however periodic is strongest against an attacker playing exponential. From the attacker's viewpoint, the exponential is strongest when the defender is playing a last move, but periodic is strongest when the defender is playing last move. Therefore, the proportion of strategies in each player population rotates dependant on which strategy is currently being played the most in the opponent's population. So in this case, there is no dominant strategy for either population.

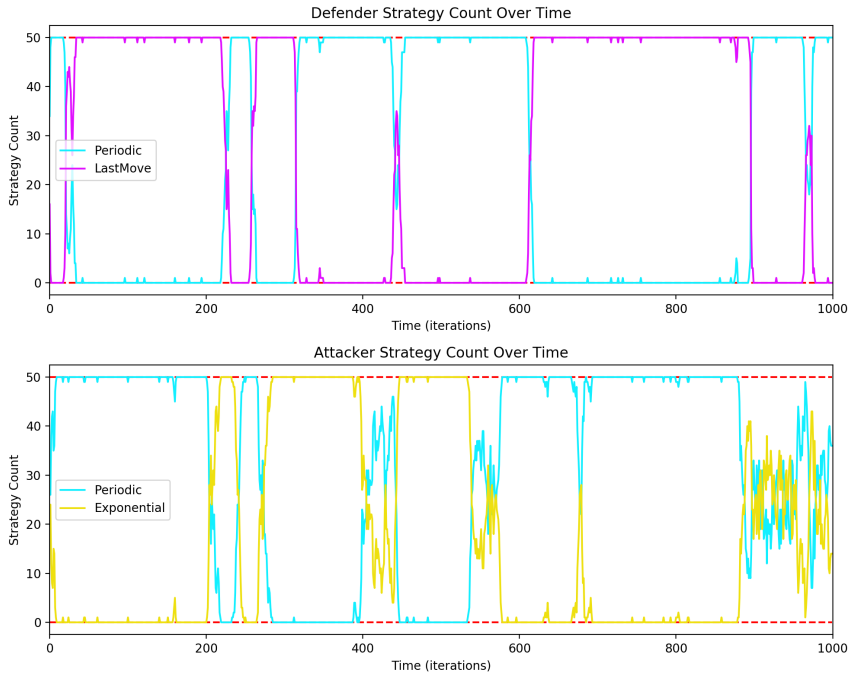


Figure 6.20: Periodic/last move defender population against a periodic/exponential attacker population

6.5 Summary

In this chapter, we completely remove the assumption that players know analytically their payoff functions and move costs. To do this we used a genetic algorithm, creating a population of strategies for both the defender and attacker. We analyse the evolution of the populations over time and find that under specific strategy classes the populations converge to analytic equilibria calculated in 4. After this, we applied our new genetic algorithm to populations containing multiple strategy classes. We studied the dominance of specific strategy classes based on the classes in opposing populations. We found that in the non-adaptive cases, the periodic strategy class comes out on top, dominating the exponential class. In the adaptive case, we find the last move strategy class is strong against periodic strategies, however very weak against the exponential strategy class.

CHAPTER 7

DISCUSSION AND FURTHER WORK

In this work, we have explored and expanded the game of FlipIt [27]. We have called this Threshold FlipThem. In doing so, we have developed a framework to analytically explore various strategies for protecting and compromising technological systems, abstracting away the cryptographic details. The framework allows for some obvious extensions, discussed here.

Reset type: In Chapter 3, we introduced the game of single-rate (n,k)-FlipThem, restricting the number of rates to just one per player, whilst keeping the game played over n resources and a threshold of k. Within Chapter 3, we also introduced two forms of reset, full reset and single reset. In full reset, the defender gains control of all resources in just one move. In single reset, the player gains control of one resource in one move. A possible extension to this could be to allow both forms of reset within one game. One could group certain resources together, perhaps to model geographic clustering, allowing a single-rate of play on a group of resources and multi-rate on individual resources.

Threshold: We have seen throughout this work that threshold is crucial to the results of the Threshold FlipThem game. So far, we have considered a *hard threshold*. By this we mean that only one player can receive benefit at a time. An example would be $n = 3, k = 2$. The attacker must have compromised at least 2 resources simultaneously to receive any gain, whilst the defender must be in control of at least 2 resources to receive any gain. Neither can happen simultaneously. As an extension to our work we could consider a *soft threshold* in which players could receive gain simultaneously in certain scenarios. For example, in the $n = 3, k = 2$ case we could assign gain to the attacker when he has compromised at least 2 resources and gain to the defender when she has compromised at least 1 resource.

This can be used to model more realistic scenarios where system architects can view the

adversary being able to steal only small amounts of information while the system is still up and running as a benefit. Likewise, the adversary could be content to be in control of just a small amount of the system, these levels of desired control can overlap. Note that both the hard threshold and soft threshold cases are already available in the Threshold FlipThem simulation lab.

Benefit variations: Our Threshold FlipThem game assumes that being in control of the system in a given interval of time is of the same value at the start of the game as in the middle of the game. Variations of this game could include making the benefit dependant on time. Imagine a scenario in which a government official has placed an important classified document on the network. As time passes, the gravity of this document being leaked could lessen with time. Thus, one could introduce some form of decay function.

Another example could be to model the cost of moving as following a one-dimensional Brownian motion. This could be used to model servers being used to trade assets. Adversaries attempting to steal assets by compromising the system may want to time the attack for when assets are at their most valuable and hence receive the most reward from their efforts.

Cost variations: Our Threshold FlipThem game assumes that the move costs c_r^i for each player i on each resource $\mathscr{R}_r$ is constant throughout the game. This could be changed so that the longer a player is in control of a resource, the higher the move cost becomes for the opponent to take back control. Another variation could be to model a "supply and demand" scenario such that the more players move within a given time interval, the more expensive move costs become.

Multiplayer games: So far we have only considered two player games. A natural extension would be to consider at least three players within the game of Threshold FlipThem. One could consider all players trying to take ownership of the resource so that only one can be in control of the resource at any one time. Or we could assume there is one defender and many attackers all attempting to compromise the system. There are of course many interesting variations to this.

Finite-time games: All mathematical analysis done in this work considers the Threshold FlipThem game over a infinite-time horizon. Whilst this is useful for analysis, this is unlike the real world. Therefore, considering finite-time, or transient, games could be useful to gain a better understanding of how best to defend systems in the short run. For smaller time horizons, optimal strategies could be computed exactly. Note that our simulations in Chapter 6 and Appendix A use a finite-time horizon.

Strategy library: The main strategies we have considered in this work are periodic, exponential and last move. All of these have been implemented in our FlipThem simulation lab. The next stage would be to implement more strategies and test them using our genetic algorithm framework. Last move is currently the only adaptive strategy we have implemented, and whilst it is useful against a periodic strategy it performs terribly against the exponential strategy class.

Therefore, adaptive strategies that perform well against multiple non-adaptive strategies could be a very interesting area of research.

More learning: So far we have considered two types of learning, fictitious play and genetic algorithms. It would be an interesting addition to the FlipThem simulation lab to include reinforcement learning (RL), assuming the game is played under a finite time horizon. Analysing how accurately RL can learn the opponent's strategy (assuming it is fixed) and therefore respond to maximise benefit. One could also put two RL strategies against each other, shedding light on new strategies that have not been considered before.

CHAPTER 8

APPENDIX A - SIMULATION LAB

In this appendix, we look at the FlipThem simulation library designed to test strategy performance. We'll begin with showing how to set up a game, run it and print the results. After this we'll see how easy it is to animate the game.

8.1 Downloading the simulation lab

First thing we require is to clone the simulation lab. This will create the directory 'FlipThem-Simulation'. The framework requires python 3.6.

```
git clone https://github.com/csherfield/FlipThem-Simulation.git
cd FlipThem-Simulation
python first_run.py
```

8.2 Running your first game

The first thing we need to do is imports. The `System` class manages the resources (or `Servers`) for the `Game` class to manipulate. The `Player` class holds the strategy classes associated to each resource. The `Exponential` class is an example of a strategy class that we use for this example. It will be shown how to generate strategy classes such as this in a later section.

```
from system import System
from game import Game
from strategies.player import Player
from strategies.server_strategies.exponential import Exponential
```

After the import we declare a `System` class. In this example we are going to be playing the game over 3 resources. After this, we define some dictionaries containing player information such as move costs and the threshold required for the player to receive any reward. We're playing over 3 resources, therefore the `move_costs` for each player properties contains a tuple of size 3. In this case, we have also set the attacker threshold to be 2. In other words, we are playing the game (3,2)-FlipThem.

```
s = System(3)
defender_properties = {'move_costs': (0.6, 0.4, 0.3), 'threshold': 2}
attacker_properties = {'move_costs': (0.3, 0.3, 0.2), 'threshold': 2}
```

Next, we define the defender and attacker with the `Player` class. This takes the `name` parameter which is a string, the `player_properties` dictionary defined earlier for defender and attacker and finally a tuple containing `strategies`. These are the strategy classes we've been discussing throughout this work. Each of the strategies takes one float representing the rate of play.

```
defender = Player(name='Defender',
                  player_properties=defender_properties,
                  strategies=(Periodic(0.5), Exponential(0.8), Periodic(0.3)))
attacker = Player(name='Attacker',
                  player_properties=attacker_properties,
                  strategies=(Exponential(0.7), Periodic(0.8), Periodic(0.7)))
```

Now, we define a dictionary containing game properties. This only contains the `time_limit` we'll be playing the game over. In this case, we choose the small value of 5 so that we can animate this comfortably later. Next, we finally define the `Game` by feeding in the two players, system and game properties. To start the game, we call the `play()` function.

```
game_properties = {'time_limit': 5}
g = Game(players=(defender, attacker),
         system=s,
         game_properties=game_properties)
g.play()
```

The game has now been played. In order to access the results, we call the method `g.print_full_game_summary()`, producing the following output:

```
    ---- System benefit ----
Defender control times: [(0.0, 0.04402745880857041),
```

```
(1.5124063117973117, 1.939735282220486), (3.3637992113235144, 3.6503334881902663)]
Defender reward: -0.22842185878030072
Attacker control times: [(0.04402745880857041, 1.5124063117973117), (1.939735282220486,
3.3637992113235144), (3.6503334881902663, 5)]
Attacker reward: 0.2784218587803006
----------------------------
```

We can see the times each player is in control of the whole system and the reward given for each player. We can also look at individual resource (server) stats by calling `g.print_individual_server_summary()`. This produces the output:

```
    ========== Server 0 ==========
Defender
Gain value: 2.2561643183820275
Number of moves: 3
Overall Benefit 0.3312328636764055
Overall Benefit [(0.0, 0.04402745880857041), (1.3637992113235147, 1.939735282220486),
(3.3637992113235144, 5)]
--------
Attacker
Gain value: 2.7438356816179725
Number of moves: 4
Overall Benefit 0.4287671363235945
Overall Benefit [(0.04402745880857041, 0.2544082859377535), (0.2544082859377535,
1.3637992113235147), (1.939735282220486, 2.847116141232217), (2.847116141232217,
3.3637992113235144)]
--------
Total time: 5.0
------------------------
========== Server 1 ==========
Defender
Gain value: 2.7239381691429116
Number of moves: 4
Overall Benefit 0.3047876338285823
Overall Benefit [(0.0, 0.9494324855118934), (0.9494324855118934, 1.1503334881902663),
(1.5124063117973117, 2.4003334881902663), (2.9646559836305753, 3.6250297048849562),
(3.6250297048849562, 3.6503334881902663)]
--------
Attacker
Gain value: 2.2760618308570884
```

```
Number of moves: 5
Overall Benefit 0.20521236617141767
Overall Benefit [(1.1503334881902663, 1.5124063117973117), (2.4003334881902663,
2.9646559836305753), (3.6503334881902663, 4.900333488190267), (4.900333488190267,
5)]
--------
Total time: 5.0
------------------------
========== Server 2 ==========
Defender
Gain value: 0.3237892986949103
Number of moves: 1
Overall Benefit 0.04475785973898206
Overall Benefit [(0.0, 0.032542385982006684), (2.5984383304119603, 2.889685243124864)]
--------
Attacker
Gain value: 4.67621070130509
Number of moves: 5
Overall Benefit 0.735242140261018
Overall Benefit [(0.032542385982006684, 1.4611138145534353), (1.4611138145534353,
2.5984383304119603), (2.889685243124864, 4.318256671696292), (4.318256671696292,
5)]
--------
Total time: 5.0
------------------------
```

Finally, we can animate the game by

```
1 a = Animate()
2 a.start(g)
```

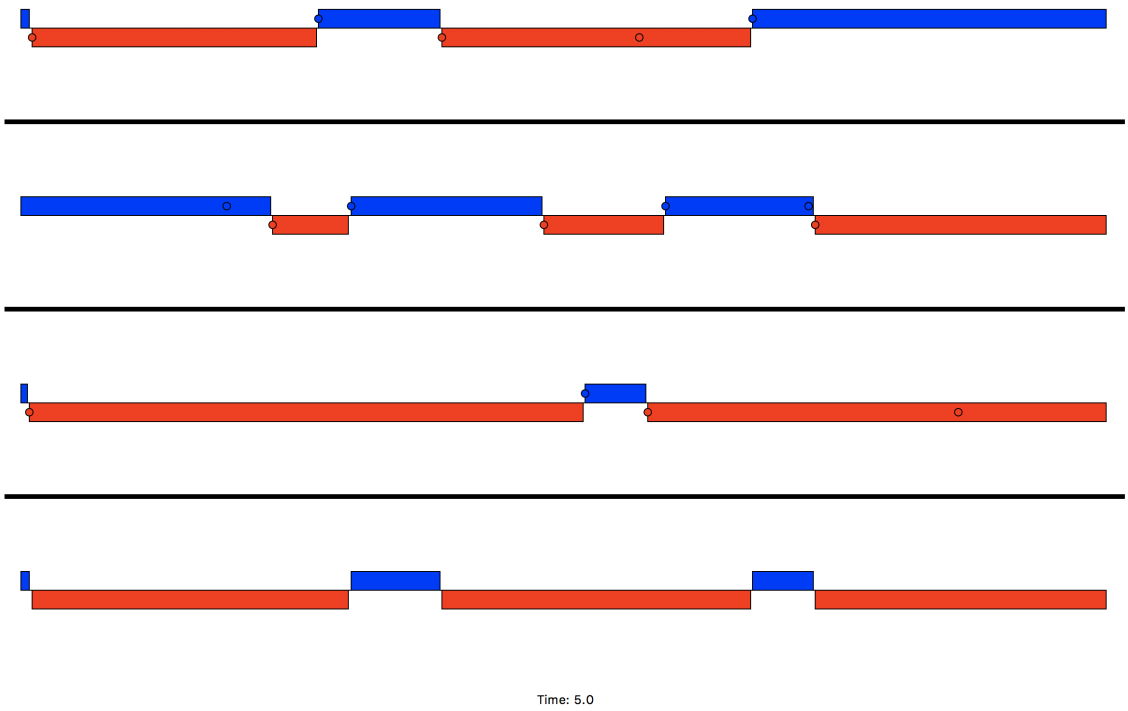


Figure 8.1: Example animated game of threshold FlipThem

8.2.1 Full script

The whole script is as follows:

```
from system import System
from game import Game
from strategies.player import Player
from strategies.server_strategies.exponential import Exponential

s = System(3)
defender_properties = {'move_costs': (0.6, 0.4, 0.3), 'threshold': 2}
attacker_properties = {'move_costs': (0.3, 0.3, 0.2), 'threshold': 2}

defender = Player(name='Defender',
                  player_properties=defender_properties,
                  strategies=(Periodic(0.5), Exponential(0.8), Periodic(0.3)))
attacker = Player(name='Attacker',
                  player_properties=attacker_properties,
                  strategies=(Exponential(0.7), Periodic(0.8), Periodic(0.7)))

game_properties = {'time_limit': 5}
g = Game(players=(defender, attacker),
```

```
19          system=s,
20          game_properties=game_properties)
21  g.play()
22
23  a = Animate()
24  a.start(g)
```

8.3 Running your first GA

Here we'll show how to set up your first genetic algorithm, testing populations full of exponential strategies and plot the results.

```
1  from genetic_algorithms.genetic_algorithm import GeneticAlgorithm
2  from strategies.server_strategies.exponential import Exponential
3  from strategies.player import Player
4  from tournament import TOURNAMENT_TYPE
```

Here, we have imported the modules required to run some form of GA. We are going to fill the two (defender and attacker) populations with exponential strategies only. For this, we have imported the `Exponential` class. We have also imported the `Player` class in order to create players to fill the population and also the `TOURNAMENT_TYPE` enumeration, to decide whether we want to run it deterministically using payoff formulae or stochastically as a simulation. In this case we will run it stochastically. Next we declare dictionaries defining the properties of the ga, game, tournament, and player.:

```
1  game_properties = {'time_limit': 200}
2  tournament_properties = {
3      'number_of_rounds': 5,
4      'attacker_threshold': 1,
5      'defender_threshold': 1,
6      'selection_ratio': 1.0,
7      'tournament_type': TOURNAMENT_TYPE.STOCHASTIC,
8      'game_properties': game_properties
9  }
```

The GA runs a tournament each generation between the two populations. The number of rounds is the number of games played between each gene (player) in the population. The defender and attacker threshold are the number of resources each player is required to be in control of for the player to receive payoff. The selection ratio is between 0 and 1 and is the proportion of matches to be played between the two populations. The tournament type is either `DETERMINISTIC` (using analytic payoffs) or `STOCHASTIC` (simulated).:

```
ga_properties = {
    'mutation_rate': 0.002,
    'file_location': 'data/stochastic/location/',
     'lower_bound': 0.0,
     'upper_bound': 3.0,
     'print_out': True
}
```

The ga_properties represent the set up of the genetic algorithm. The mutation_rate is how often a gene is likely to be mutated. The file_location is where all data is to be written. The upper_bound is the highest rate that can be introduced to the population. Note that in the stochastic case, having a higher upper bound slows down the simulations. print_out is a boolean, if true it will print out more information to the console.

```
defender_ga_properties = {
    'name': "Defender ",
    'number_of_players': 50,
    'strategy_classes': (Exponential,),
    'move_costs': (0.2,),
}
attacker_ga_properties = {
    'name': "Attacker ",
    'number_of_players': 50,
    'strategy_classes': (Exponential,),
    'move_costs': (0.3,),
}
```

Above, we define the properties for the defender population and the attacker population. name is purely the name of population we're referring to, plus a number for each gene in the population. number_of_players is the size of the population. stategy_classes is a tuple of strategies, taken from the strategies.server_strategies directory. In this case we only include Exponential, however if desired we can put in any of the current strategies available. move_costs is a tuple of costs, one for each server. In this case it is a tuple of length as we are only playing over one resource.

```
ga = GeneticAlgorithm(defenders=defender_ga_properties,
                      attackers=attacker_ga_properties,
                      ga_properties=ga_properties,
                      tournament_properties=tournament_properties)
ga.run(5000, 100)
ga.plot()
```

Finally, we can call the `GeneticAlgorithm` class with all the dictionaries defined above. After this we call the `ga.run()` method for 5000 iterations, writing to file every 100 iterations. Once this is complete, we plot the results using `ga.plot()`.

8.3.1 Full script

The whole script is as follows:

```
from genetic_algorithms.genetic_algorithm import GeneticAlgorithm
from strategies.server_strategies.exponential import Exponential
from strategies.player import Player
from tournament import TOURNAMENT_TYPE

game_properties = {'time_limit': 200}
tournament_properties = {
    'number_of_rounds': 5,
    'attacker_threshold': 1,
    'defender_threshold': 1,
    'selection_ratio': 1.0,
    'tournament_type': TOURNAMENT_TYPE.STOCHASTIC,
    'game_properties': game_properties
}

ga_properties = {
    'mutation_rate': 0.002,
    'file_location': 'data/stochastic/location/',
     'lower_bound': 0.0,
     'upper_bound': 3.0,
     'print_out': True
}

defender_ga_properties = {
    'name': "Defender ",
    'number_of_players': 50,
    'strategy_classes': (Exponential,),
    'move_costs': (0.2,),
}
attacker_ga_properties = {
    'name': "Attacker ",
    'number_of_players': 50,
    'strategy_classes': (Exponential,),
    'move_costs': (0.3,),
}

ga = GeneticAlgorithm(defenders=defender_ga_properties,
                      attackers=attacker_ga_properties,
                      ga_properties=ga_properties,
```

```
40                          tournament_properties=tournament_properties)
41 ga.run(5000, 100)
42 ga.plot()
```

8.4 Performance notes

All simulations were run on a 2016 MacBook Pro with specifications:

- Processor: 2 GHz Intel Core i5
- Memory: 16 GB 1867 MHz LPDDR3

The script above takes on this machine (on average) 53 minutes to run.

BIBLIOGRAPHY

[1] H. S. BEDI, S. G. SHIVA, AND S. ROY, *A game inspired defense mechanism against distributed denial of service attacks*, Security and Communication Networks, 7 (2014), pp. 2389–2404.

[2] M. BENAIM AND M. W. HIRSCH, *Mixed equilibria and dynamical systems arising from fictitious play in perturbed games*, Games and Economic Behaviour, 29 (1999), pp. 36–72.

[3] K. D. BOWERS, M. VAN DIJK, R. GRIFFIN, A. JUELS, A. OPREA, R. L. RIVEST, AND N. TRIANDOPOULOS, *Defending against the unknown enemy: Applying flipit to system security*, in Grossklags and Walrand [10], pp. 248–263.

[4] G. BROWN, *Iterative solution of games by fictitious play*, Koopmans, T.C. (Ed.), Activity Analysis of Production and Allocation, (1951), pp. 374–376.

[5] H. ÇEKER, J. ZHUANG, S. J. UPADHYAYA, Q. D. LA, AND B. SOONG, *Deception-based game theoretical approach to mitigate DoS attacks*, in Zhu et al. [31], pp. 18–38.

[6] W. FELLER, *An Introduction to Probability Theory and Its Applications*, vol. 1, Wiley, January 1968.

[7] D. FUDENBERG AND D. KREPS, *Learning mixed equilibria*, Games and Economic Behavior, 5 (1993), pp. 320–367.

[8] D. FUDENBERG AND D. K. LEVINE, *The Theory of Learning in Games*, The MIT Press, 1st edition ed., 1998.

[9] G. GRIMMETT AND D. STIRZAKER, *Probability and Random Processes*, Oxford University Press, 3rd edition ed., 2001.

[10] J. GROSSKLAGS AND J. C. WALRAND, eds., *Decision and Game Theory for Security - Third International Conference, GameSec 2012, Budapest, Hungary, November 5-6, 2012. Proceedings*, vol. 7638 of Lecture Notes in Computer Science, Springer, 2012.

[11] C. HERLEY, *The plight of the targeted attacker in a world of scale.*, in WEIS, 2010.

[12] J. H. HOLLAND, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, MIT Press, Cambridge, MA, USA, 1992.

[13] A. R. HOTA, A. A. CLEMENTS, S. SUNDARAM, AND S. BAGCHI, *Optimal and game-theoretic deployment of security investments in interdependent assets*, in Zhu et al. [31], pp. 101–113.

[14] A. LASZKA, G. HORVATH, M. FELEGYHAZI, AND L. BUTTYAN, *Flipthem: Modeling targeted attacks with flipit for multiple resources*, in Poovendran and Saad [24], pp. 175–194.

[15] A. LASZKA, B. JOHNSON, AND J. GROSSKLAGS, *Mitigation of Targeted and Non-targeted Covert Attacks as a Timing Game*, Springer International Publishing, Cham, 2013, pp. 175–191.

[16] D. LESLIE, C. SHERFIELD, AND N. P. SMART, *Threshold FlipThem: When the winner does not need to take all*, in Decision and Game Theory for Security - 6th International Conference, GameSec 2015, London, UK, November 4-5, 2015, Proceedings, M. H. R. Khouzani, E. Panaousis, and G. Theodorakopoulos, eds., vol. 9406 of Lecture Notes in Computer Science, Springer, 2015, pp. 74–92.

[17] ——, *Multi-rate threshold flipthem*, in Computer Security – ESORICS 2017, S. N. Foley, D. Gollmann, and E. Snekkenes, eds., Cham, 2017, Springer International Publishing, pp. 174–190.

[18] D. S. LESLIE AND E. J. COLLINS, *Generalized weakened fictitious play*, Games and Economic Behavior, 56 (2006), pp. 285–298.

[19] B. Z. MOAYEDI AND M. A. AZGOMI, *A game theoretic framework for evaluation of the impacts of hackers diversity on security measures*, Rel. Eng. & Sys. Safety, 99 (2012), pp. 45–54.

[20] J. NASH, *Non-cooperative games*, The Annals of Mathematics, 54 (1951), pp. 286–295.

[21] M. OSBORNE AND A. RUBINSTEIN, *A Course in Game Theory*, MIT Press, 1994.

[22] E. PANAOUSIS, A. FIELDER, P. MALACARIA, C. HANKIN, AND F. SMERALDI, *Cybersecurity games and investments: A decision support approach*, in Poovendran and Saad [24], pp. 266–286.

[23] V. PHAM AND C. CID, *Are we compromised? modelling security assessment games*, in Grossklags and Walrand [10], pp. 234–247.

[24] R. POOVENDRAN AND W. SAAD, eds., *Decision and Game Theory for Security - 5th International Conference, GameSec 2014, Los Angeles, CA, USA, November 6-7, 2014. Proceedings*, vol. 8840 of Lecture Notes in Computer Science, Springer, 2014.

[25] S. RASS AND Q. ZHU, *GADAPT: A sequential game-theoretic framework for designing defense-in-depth strategies against advanced persistent threats*, in Zhu et al. [31], pp. 314–326.

[26] M. SMYRNAKIS AND D. S. LESLIE, *Dynamic opponent modelling in fictitious play*, The Computer Journal, 53 (2010), pp. 1344–1359.

[27] M. VAN DIJK, A. JUELS, A. OPREA, AND R. L. RIVEST, *Flipit: The game of "stealthy takeover"*, J. Cryptology, 26 (2013), pp. 655–713.

[28] M. P. WELLMAN AND A. PRAKASH, *Empirical game-theoretic analysis of an adaptive cyber-defense scenario (preliminary report)*, in Poovendran and Saad [24], pp. 43–58.

[29] M. ZHANG, Z. ZHENG, AND N. B. SHROFF, *A game theoretic model for defending against stealthy attacks with limited resources*, CoRR, abs/1508.01950 (2015).

[30] Z. ZHOU, N. BAMBOS, AND P. W. GLYNN, *Dynamics on linear influence network games under stochastic environments*, in Zhu et al. [31], pp. 114–126.

[31] Q. ZHU, T. ALPCAN, E. A. PANAOUSIS, M. TAMBE, AND W. CASEY, eds., *Decision and Game Theory for Security - 7th International Conference, GameSec 2016, New York, NY, USA, November 2-4, 2016, Proceedings*, vol. 9996 of Lecture Notes in Computer Science, Springer, 2016.